# A simulation study to assess the impact of various measurement error characteristics on parameter estimation in the Cox model - an application to the Wismut cohort

Bachelor Thesis


**Author**: Juliana Schäfer
**Supervisors**: Dr. Sabine Hoffmann, Nicole Schüller

Department of Statistics
Faculty of Mathematics, Computer Science and Statistics
Ludwig-Maximilians University of Munich

September 18, 2019

# Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht.
Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

**Juliana Schäfer**

München, den 19.09.2019

# Abstract

In the Wismut cohort, an occupational cohort of uranium miners, where are uncertainties in the radon exposure assessment. The aim of this thesis is to investigate the effects on parameter estimates in the Cox model that may arise as a result. Various simulation studies are performed for this purpose.

First, the effect of the rounding of values on the estimate is examined. Subsequently, the assignment of exposure values based on the group affiliation of the workers is examined. The procedure is then analyzed, in which individual measurements are used to estimate the exposure values of an object. This is also considered in combination with the previous measurement error. Finally, this thesis deals with the uncertainty caused by expert estimates.

For this approach, a fictive data set of the Wismut cohort, as well as a self-generated data set, are used. In addition, various measurement error characteristics are applied, such as unshared and shared errors, homoscedastic and heteroscedastic errors, as well as various variances.

# Contents

# List of Figures

# List of Tables

# List of Symbols

| | |
|---|---|
| $i$ | Individual (miner) |
| $t$ | Time, year |
| $j$ | Group, e.g. activity or shaft, $j = 1, ..., J$ |
| $o$ | Object |
| $X$ | Real radon exposure |
| $X^{cum}$ | Cumulative real radon exposure |
| $Z$ | Observed radon exposure |
| $Z^{cum}$ | Cumulative observed radon exposure |
| $U$ | Error |
| $g(t, o)$ | Equilibrium factor |
| $c(o)$ | Correction factor |
| $f(t, o, j)$ | Activity weighting factor |
| $w(t, o)$ | Working time factor |
| $\bar{\bar{C}}_{RDP}(t, o)$ | Mean radon progeny concentration |
| $\bar{\bar{C}}_{Rn}(t, o)$ | Mean radon gas concentration |
| $\mathcal{M}$ | Model |

# 1 Introduction

The Wismut cohort deals with former miners in the GDR and investigates the influence of radon exposure and resulting diseases. Thereby there are various sources of uncertainty that may affect the assessment of radon influence. The following ones are discussed in this thesis:

- Due to missing measurements, exposure values were estimated by experts.
- Individual miner exposure values were estimated by general measurements.
- Individual measurements were used to estimate the exposure values of a whole object.

For this purpose, simulation studies are performed for individual errors and error combinations. Here Cox models and the possible bias of coefficient estimators are considered. Since the exposure values of the workers differ from year to year, a Cox model with time-dependent covariates is used for the investigations.

The problem of measurement errors may also be transferred to other epidemiological studies, in particular occupational cohort studies.

In this thesis, first, an overview of the Wismut cohort and its exposure assessment is given. Then the theory about the Cox model with time-dependent covariates and measurement errors is covered. A fictive data set of the Wismut cohort and a self-generated data set are used for the simulation studies. The approach is described in more detail, before the actual simulated models and their results are described.

# 2 The Wismut cohort

This section gives an overview of the Wismut cohort and the exposure assessment. It is based on (Kreuzer, Schnelzer, Tschense, Walsh, & Grosche, 2009) and (Küchenhoff et al., 2018).

After the 2nd World War, old silver mines in the former German Democratic Republic (GDR) were reopened to produce uranium for Soviet nuclear weapons under the name Wismut. Especially in the early years (1946-1955), the working conditions were very bad. There were no safety measures and protection against radiation and dust. Therefore, the workers were exposed to high levels of exposure. After the first radon measurements were made in 1955, the working conditions improved in the following years. The number of workers was reduced from about 100 000 to 30 000 - 40 000 and forced ventilation and wet drilling were introduced. Stricter international guidelines were introduced in 1970. As a result, the number of workers was further reduced to 20 000 and working conditions and safety precautions were improved.

The Wismut cohort includes about 59 000 former workers and deals, among other things, with the development of lung cancer due to radon exposure. The estimation of radon exposure was based on a job-exposure matrix (JEM). It contains exposure values in working level months (WLM) per year, job and workplace. 1 WLM corresponds to $1.3 \times 10^5$ MeV of alpha energy/l air by radon progenies for 170 hours.
Since no radon measurements were available for the years before 1955, the values for this period were estimated by experts. Thereby factors such as ventilation were considered.
For the JEM the arithmetic means of the radon concentration measurements $\bar{\bar{C}}_{Rn}(t, o)$ (in $kBq/m^3$) and the radon progeny measurements $\bar{\bar{C}}_{RDP}(t, o)$ (in $MeV/cm^3$) were used and converted into WLM. They were then adjusted by a working time factor and an activity factor. In total, the exposure values were calculated by $\bar{\bar{C}}_{Rn}(t, o) \cdot 12 \cdot f(t, o, j) \cdot w(t, o) \cdot g(t, o)$ and $\bar{\bar{C}}_{RDP}(t, o) \cdot 12 \cdot f(t, o, j) \cdot w(t, o) \cdot c(t, o)$ respectively.

The individual factors remain as follows:

- The equilibrium factor $g(t, o)$ depends on the air exchange and corrects the radon gas concentration measurements in this regard. It has been determined per object and year by experts and depends on the ventilation conditions. The smaller $g(o, t)$, the better the ventilation. It varies between 0.2 and 0.6. As ventilation has improved over time, the values are lower in

late years.

- The correction factor $c(o)$ is used for radon progeny concentration measurements to include deficits and disruptions in ventilation. It ranges from 1.2 to 1.45.
- The activity weighting factor $f(t, o, j)$ lies between 0 and 1 depending on the activity performed by a miner.
- The working time factor $w(t, o)$ adjusts the working time of the hewers. This has decreased over the years, so the factor also reduces over time from 1.2 to 0.88.

# 3 Methods

## 3.1 The Cox proportional-hazards model with time-dependent covariates

To provide an overview of the theory behind the initial models used, a short introduction to the Cox proportional-hazard model with time-dependent covariates is given. Reference is made to Cox (1972), Hendry (2014), Kleinbaum and Klein (2012) and Fahrmeir (2007).

The Cox proportional-hazards model is one of the most widely used models in survival analysis. If the covariates are not fixed but change over time, the model can be extended to the Cox model with time-dependent covariates. The hazard rate of subject $i$ experiencing the event of interest is as follows:

$$\lambda_i(t) = \lambda_0(t) \cdot exp(X_i(t)\beta) \tag{1}$$

with the covariates for subject $i$ $X_i(t) \in \mathbb{R}^{1xp}$ that can be either time-independent or time-dependent and the coefficients $\beta \in \mathbb{R}^{px1}$ ($p \in \mathbb{N}$). If all covariates are equal to 0, $exp(X_i(t)\beta) = 1$ and thus $\lambda_0(t)$, the so-called baseline hazard, is the only term left. This is why $\lambda_0(t)$ can usually be regarded as the hazard rate for a subject if all covariates are equal to 0. It should also be noted, that $\lambda_0(t)$ is independent from $X_i(t) \forall i$. For two observations $i$ and $j$, it is assumed that the following relationship applies between their hazard rates:

$$\frac{\lambda_0(t)exp(X_i(t)\beta)}{\lambda_0(t)exp(X_j(t)\beta)} = \frac{exp(X_i(t)\beta)}{exp(X_j(t)\beta)}. \tag{2}$$

This means that the relative effects between two observations can be expressed by $\beta$ alone. Thus the hazard ratio between a subject with covariates $X_i(t)$ and a subject with all covariates equal to 0, can be represented by $exp(X_i(t)\beta)$. The estimation of $\beta$ is then done, similar to the simple Cox model, by maximizing the partial likelihood, assuming that $x(t)$ is a predictable stochastic process:

$$PL(\beta) = \prod_{i=1}^{k} \frac{exp(X_{(i)}(t_{(i)})\beta)}{\sum_{j \in R_{t_{(i)}}} exp(X_j(t_{(i)})\beta)}, \tag{3}$$

where $t_{(1)} < ... < t_{(i)} < ... < t_{(k)}$ ($k \leq n$) are the durations of individuals that are not yet censored and $R(t)$ is the number of individuals at risk.

Hendry (2014) shows that it is possible to use truncated piecewise exponentials for generating data that follow a Cox model with time-dependent covariates.

## 3.2 Measurement errors

Before coming to the simulation studies themselves, this chapter discusses the types of measurement error covered in this paper. First in 3.2.1, various types of errors are described in general terms. Here Hoffmann's (2017) definitions are used. Subsequently, in 3.2.2, the error types which will be later simulated are concretely addressed. This chapter is based on Küchenhoff et al. (2018), whereby further the notation of Hoffmann (2017) is used.

### 3.2.1 General classifications

#### Berkson and classical measurement error

Depending on the relationship between measurement error and observed or actual exposure and their independence assumptions, measurement errors can be divided into classical errors and Berkson errors. The term classical error is used if the measurement error of a miner $i$ at a time $t$ is independent of his actual radon exposure at this time, i.e.

$$U_i(t) \perp\!\!\!\perp X_i(t). \tag{4}$$

The corresponding observed radon exposure $Z_i(t)$ can then be modeled by $X_i(t)$ and $U_i(t)$. This means, depending on whether it is a multiplicative or an additive measurement error, $U_i(t)$ describes either the ratio between $X_i(t)$ and $Z_i(t)$ or the difference between $X_i(t)$ and $Z_i(t)$:

$$Z_i(t) = X_i(t) \cdot U_i(t) \qquad \text{resp.} \qquad Z_i(t) = X_i(t) + U_i(t). \tag{5}$$

If, however, the measurement errors are not independent of the true exposure values but the observed ones, i.e.

$$U_i(t) \perp\!\!\!\perp Z_i(t), \tag{6}$$

the error is called a Berkson error. In this case, the modeling is done exactly the other way round, i.e. the true exposure results from the observed exposure and the error. Analogous to the classical error, there is a multiplicative and an additive measurement error:

$$X_i(t) = Z_i(t) \cdot U_i(t) \qquad \text{and} \qquad X_i(t) = Z_i(t) + U_i(t). \tag{7}$$

#### Heteroscedastic and homoscedastic measurement error

Heteroscedastic errors are defined as those that have a different dispersion. Con-

cerning all miners, this means with $Var(U_i(t)) = \sigma_i^2$ standing for the measurement error variance of miner $i$, that $\sigma_i^2 \neq \sigma_{i'}^2 \ \forall i \neq i'$. On the other hand, if the variance does not differ, we speak of homoscedastic errors, i.e. $\sigma_i^2 = \sigma_{i'}^2 \ \forall i, i'$. Analogously, the terms heteroscedasticity and homoscedasticity can also be used concerning time and place.

#### Shared and unshared measurement error

If the measurement errors of the workers at a time $t$ are independent of each other ($U_i(t) \perp\!\!\!\perp U_{i'}(t) \ \forall i \neq i'$) and the errors of a miner $i$ at different moments are also independent of each other ($U_i(t) \perp\!\!\!\perp U_i(t') \ \forall t \neq t'$), then we speak of an unshared measurement error. Otherwise, the error is called a shared measurement error. It can be thereby, for example, shared between all workers at time $t$ ($U_i(t) = U(t) \ \forall i$), or within a worker $i$ for the entire working time ($U_i(t) = U_i \ \forall t$). It is also of course possible that the measurement error is only shared for subgroups of workers or only for several years.

### 3.2.2 Special measurement error types

#### Approximation error due to rounding

Perhaps the easiest error to understand is the approximation error due to rounding. It occurs when rounded values are used instead of the actual values.

#### Assignment error

An assignment error occurs when Radon Exposure values are assigned to a miner based on its membership to a group. These can be occupational and activity groups, or also an affiliation, for example, to a mine or a shaft. As already described in section 2, there is a so-called job-exposure-matrix (JEM), which indicates the radon values for different groups, which are assigned to the corresponding workers. Since the true radon exposure of a worker $i$ in group $j$ at time $t$ results from the assigned exposure $Z_{ij}(t)$ with a deviation $U_{ij}(t)$, this is a Berkson error. More precisely, the true value is obtained as follows:

$$X_{ij}(t) = Z_{ij}(t) \cdot U_{ij}(t) \qquad \text{resp.} \qquad X_{ij}(t) = Z_{ij}(t) + U_{ij}(t) \tag{8}$$

whereby, depending on the error type, $Z_{ij}(t)$ and $U_{ij}(t)$ can be the same for several groups, workers or times.
If the exposure is assigned due to shaft or mine associations, it is referred to as a spatial assignment error. Here spatial differences between the individual workers are ignored. Therefore, local radon concentrations may cause the actual radon

exposure to deviate from the assigned value.
If, on the other hand, the activity is decisive for the exposure assigned to a worker, the error is referred to as an activity assignment error. Despite the same job, the radon concentration may vary between the miners and thus differ from the assigned value.

### Generalization error

Another error that can occur in cohort studies is the generalization error. It can arise when the mean of individual measurements is used to determine the exposure in the shaft. Analogous to the assignment error, the spatial and activity variants are available here as well. The spatial generalization error uses the measurements from some locations in an object to estimate the radon concentration of the entire one. The observed value of a location $j$ of an object $o$ at time $t$ results as follows:

$$Z_{jo}(t) = X_{jo}(t) \cdot U_j(t) \qquad \text{resp.} \qquad Z_{jo}(t) = X_{jo}(t) + U_j(t). \tag{9}$$

The estimated value of the object $o$ at time $t$ is then:

$$\bar{Z}_o(t) = \frac{1}{J} \sum_{j=1}^{J} Z_{jo}(t) \tag{10}$$

where $J$ is the number of locations in object $o$. This is a classical measurement error.

### Estimation error - parameter uncertainties

As already described in section 2, radon exposure in the Wismut cohort is calculated by radon gas concentration measurements or radon progeny measurements and factors determined by experts:

$$X_i(t) = \bar{\bar{C}}_{Rn}(t, o) \cdot 12 \cdot f(t, o, j) \cdot w(t, o) \cdot g(t, o) \tag{11}$$

$$X_i(t) = \bar{\bar{C}}_{RDP}(t, o) \cdot 12 \cdot f(t, o, j) \cdot w(t, o) \cdot c(t, o). \tag{12}$$

The observed exposure values then result from:

$$Z_i(t) = \bar{\bar{C}}_{Rn}(t, o) \cdot 12 \cdot \hat{f}(t, o, j) \cdot \hat{w}(t, o) \cdot \hat{g}(t, o) \tag{13}$$

$$Z_i(t) = \bar{\bar{C}}_{RDP}(t, o) \cdot 12 \cdot \hat{f}(t, o, j) \cdot \hat{w}(t, o) \cdot \hat{c}(t, o). \tag{14}$$

The uncertainty here arises from the factors that are estimated by experts.

# 4 Simulation studies

## 4.1 Simulation design

For the simulation studies, fictive data of the Wismut cohort, which were provided by the Bundesamt für Strahlenschutz, are used as well as an own data set which is generated. First, the fictive data set (data1) is described in chapter 4.1.1, then chapter 4.1.2 shows how the new data set (data2) is created.

### 4.1.1 Fictive data of the Wismut cohort

Table 1 shows a sample of the data set, omitting unused columns. The meanings of the individual columns are explained in table 2. For a better comparison with the self-generated data (data2), figures 1, 2 and 3 show the distributions of the age at which the miners start to work, the duration of employment and the annual radon exposure. A direct comparison can also be found in the appendix (A.3).

| ID | yyborn | yyin | yystop | w46 | w47 | w48 | w49 | w50 | ... | w89 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1946 | 1974 | 1989 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 1.20 |
| 2 | 1927 | 1949 | 1969 | 0.00 | 0.00 | 0.00 | 4.20 | 8.80 | ... | 0.00 |
| 3 | 1934 | 1954 | 1958 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 0.00 |
| 4 | 1962 | 1985 | 1989 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 1.60 |
| 5 | 1908 | 1949 | 1962 | 0.00 | 0.00 | 0.00 | 18.30 | 17.90 | ... | 0.00 |
| 6 | 1918 | 1950 | 1962 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 0.00 |
| 7 | 1918 | 1949 | 1957 | 0.00 | 0.00 | 0.00 | 15.70 | 32.20 | ... | 0.00 |
| 8 | 1955 | 1975 | 1988 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 0.00 |
| 9 | 1944 | 1965 | 1989 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 0.10 |
| 10 | 1959 | 1978 | 1989 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | ... | 0.50 |

Table 1: Extract from the data set (used columns only)

| Explanations of the variables: | |
|---|---|
| ID: | miner ID |
| yyborn: | year of birth |
| yyin: | year the miner started working |
| yystop: | year the miner stopped working |
| w46 - w89: | Radon exposure in the corresponding year in WLM |

Table 2: Explanation of the variables

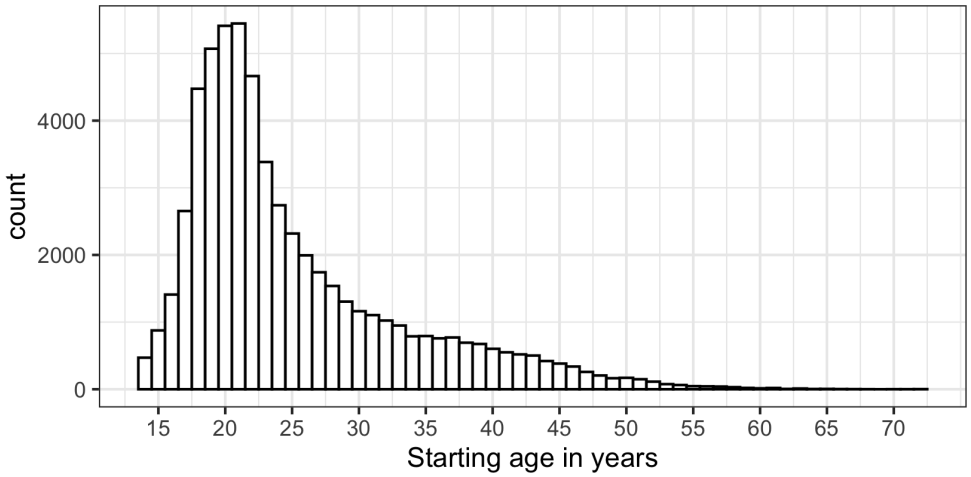


Figure 1: Starting age distribution

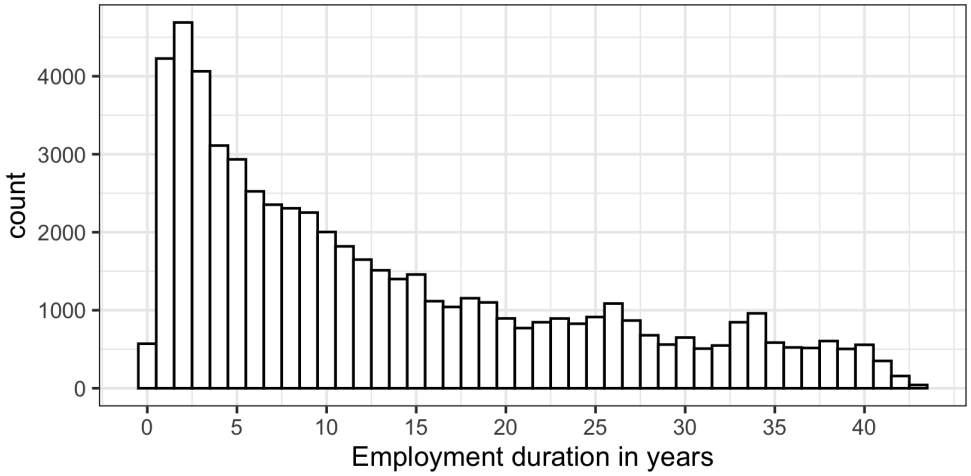


Figure 2: Employment duration distribution

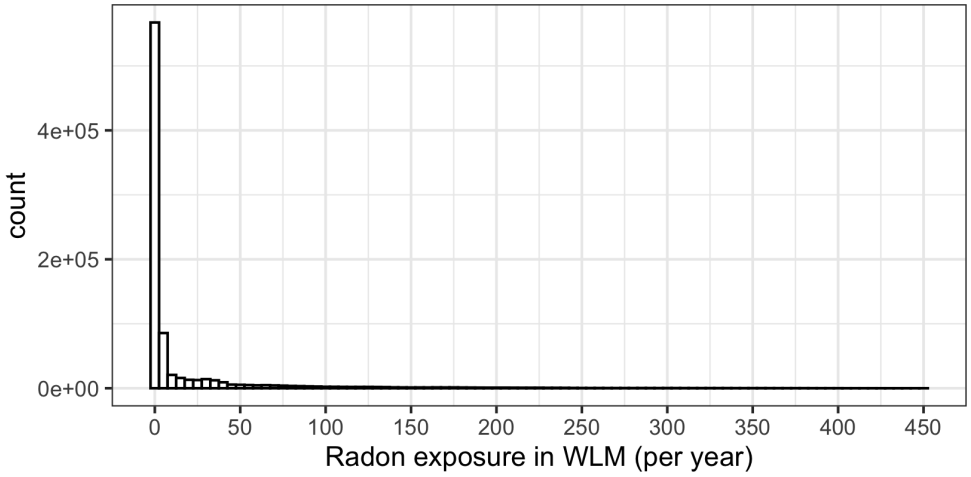


Figure 3: Radon exposure distribution

### 4.1.2 Generated data

In order to generate the data for the simulation studies to be as realistic as possible, the values of the Wismut cohort are used as a template. The data descriptions of Kreuzer et al. (2009) are adopted for this purpose.

#### Time period

Analogous to the actual opening time of the mines from the Wismut cohort (1946 - 1990), it is also only possible to work during this period in the simulation studies (see also paragraph Duration of employment). Due to simplicity, it is assumed that every worker starts in 1946.

#### Starting age

The age at which the miners started working was between 13 and 68 years, with an average age of 24 years. When generating the initial age, a log-normal distribution is used to generate values similar to the real ones in the Wismut cohort. However, only integer values in the range 15 - 65 years are used in order to omit too extreme values. Thus, values are sampled from the resulting distribution, which is presented in figure 4.

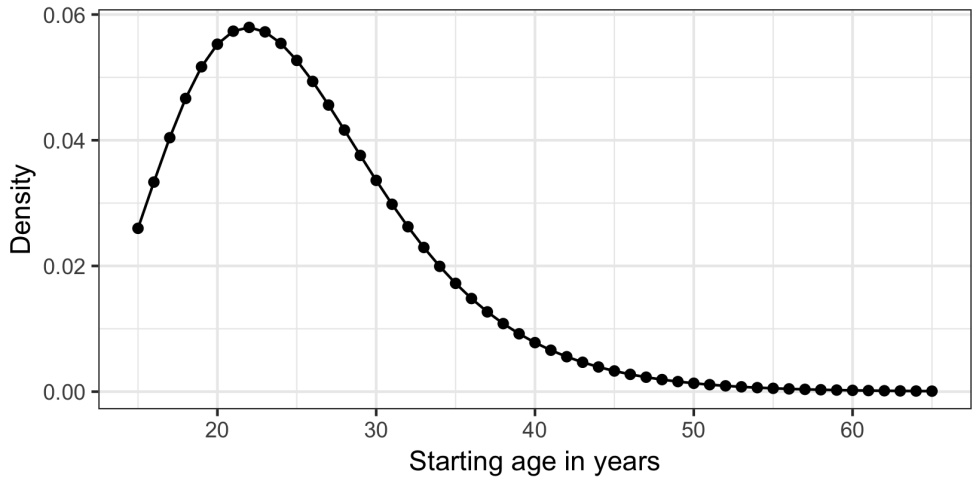


Figure 4: Starting age distribution

#### Duration of employment

On average, the miners worked for 14 years. The actual working time varied between 0.5 and 44 years. Since only annual exposures are used in the simulation studies for simplification purposes, the duration here is between 1 and 44 years. The probabilities are increased for a duration between 7 and 17 years, as these were the mean values in the individual phases of the cohort study. Thus, the

following log-normal distribution is used to generate a duration similar to the real values. Analogous to the generation of the starting age, only integer values are used. This yields the distribution in figure 5. Moreover, later, the maximum working age is set to 75 years in order not to generate unrealistically old workers. Also, it is redrawn in case the year 1990 is exceeded.

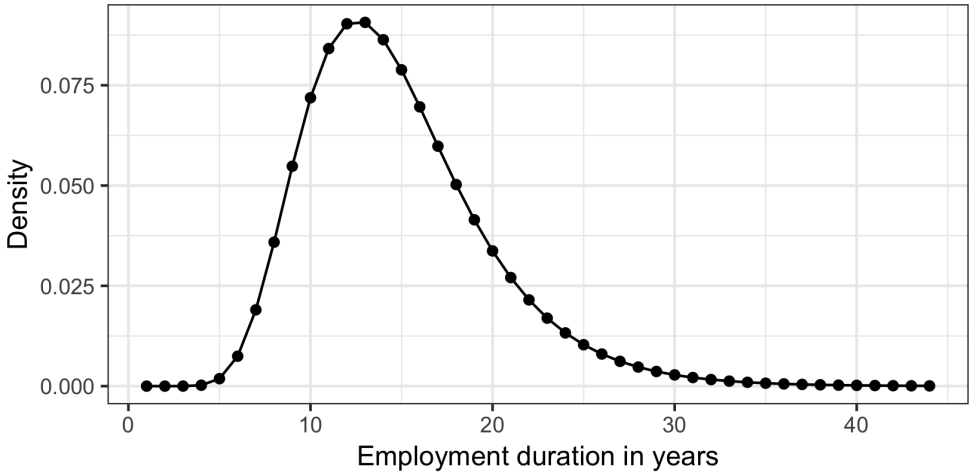


Figure 5: Employment duration distribution

### Radon exposure

The observed cumulative radon load of a miner was between 0 and 120 WLM (working level months) per year. The approximate annual exposure values are used to create a density which can be drawn from (figure 6). This distribution is used to generate both the observed and the actual exposure.

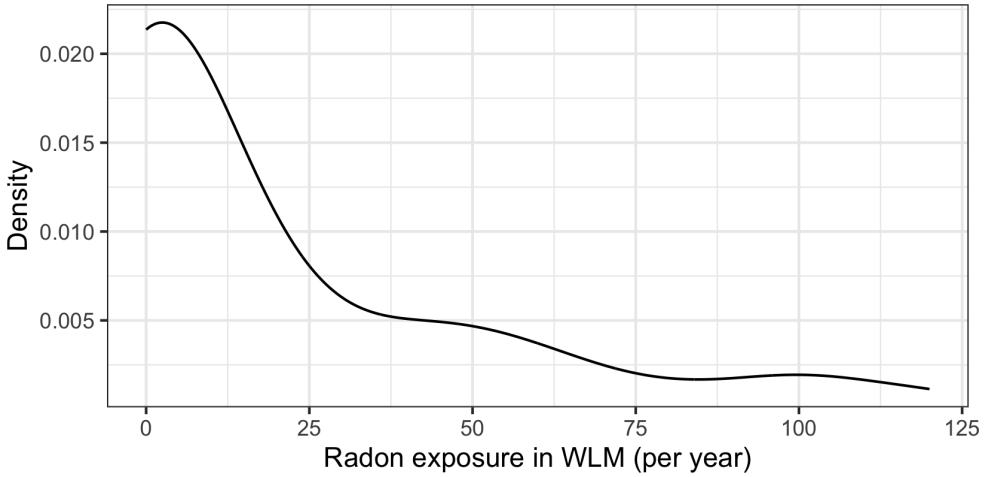


Figure 6: Radon exposure distribution

### Radon concentration measurements

The mean radon gas concentration in 1955-1957 were between 0.59 and 19 $kBq/m^3$ (Küchenhoff et al., 2018, p.59). These values were used to create a density which can be sampled from figure 7.

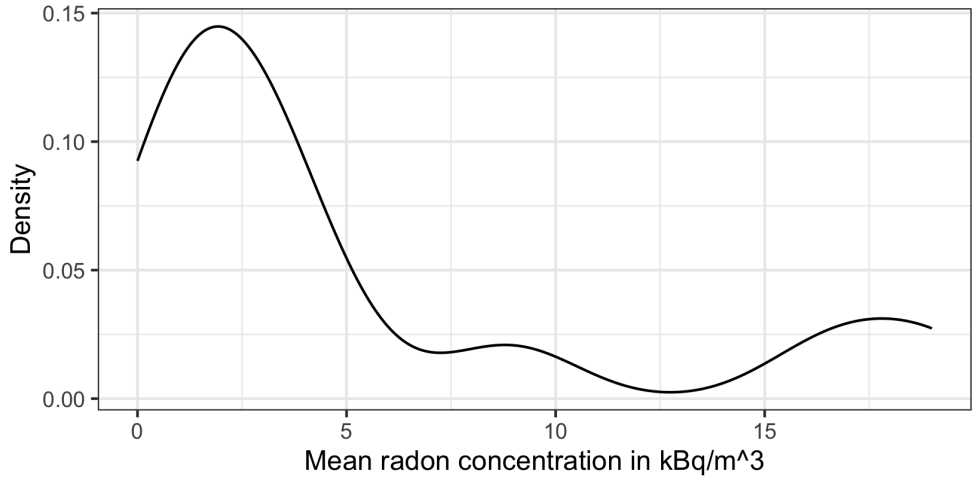


Figure 7: Mean radon concentration distribution

### Radon progeny measurements

For the distribution of the mean radon progeny the exposure values in WLM were converted to $MeV/cm^3$. For this purpose, the equation $\bar{\bar{C}}_{RDP}(t,o) \cdot 12 \cdot f(t,o,j) \cdot w(t,o) \cdot c(t,o)$ was solved for $\bar{\bar{C}}_{RDP}(t,o)$ by replacing the factors by their mean values. The distribution in figure 8 was created from the results.

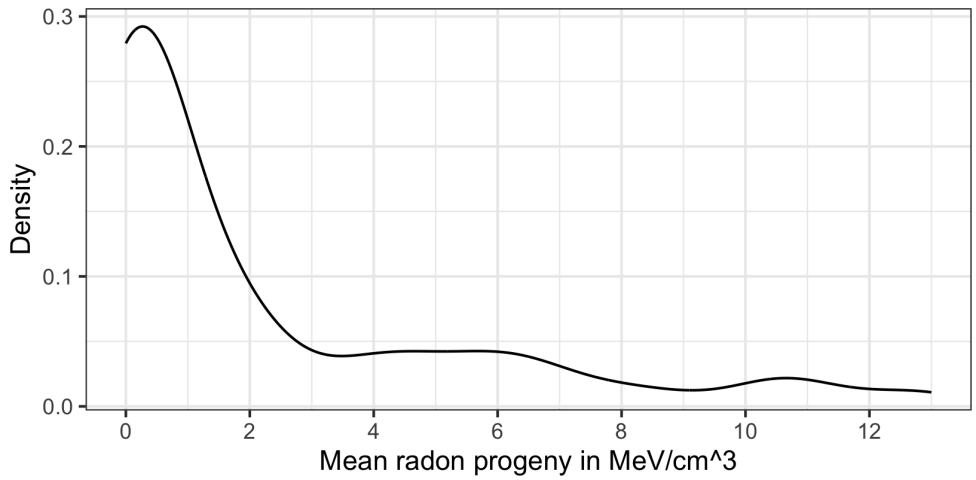


Figure 8: Mean radon progeny distribution

### Factors

The distributions of factors for the estimation error simulation studies are all based on the values that can be found in Küchenhoff et al. (2018).

The activity weighting factor $f(t, o, j)$ was between 0 and 1. More precise values for some activities were given in Küchenhoff et al. (2018, p.107), which are used to sample from (figure 9).

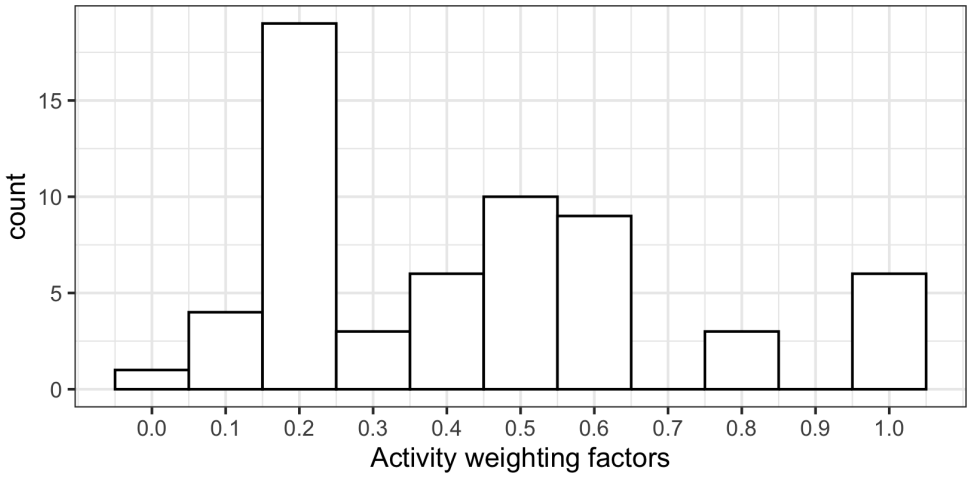


Figure 9: Activity weighting factors distribution

Values from the Wismut cohort are also used for the correction factor (Küchenhoff et al., 2018, p.53):

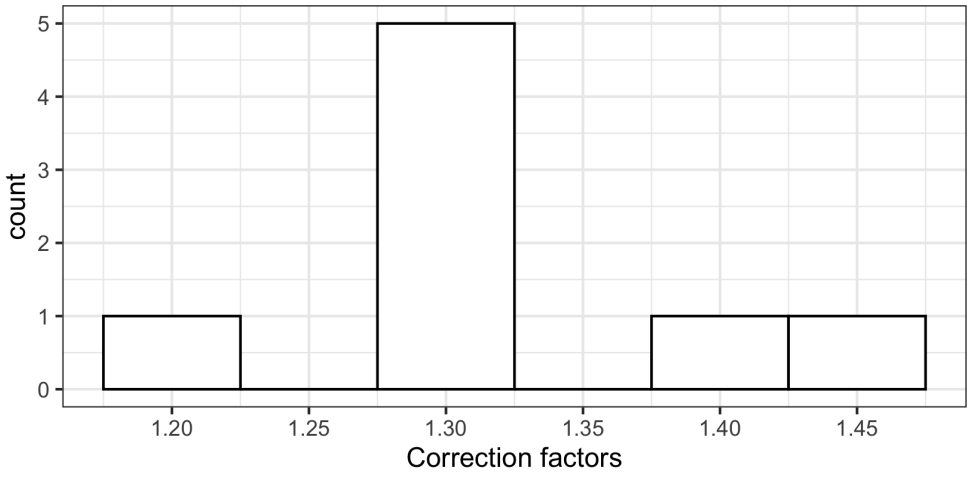


Figure 10: Correction factors distribution

The equilibrium factor was between 0.2 and 0.6. The values from Küchenhoff et al. (2018, p.45) are used for this distribution (figure 11).

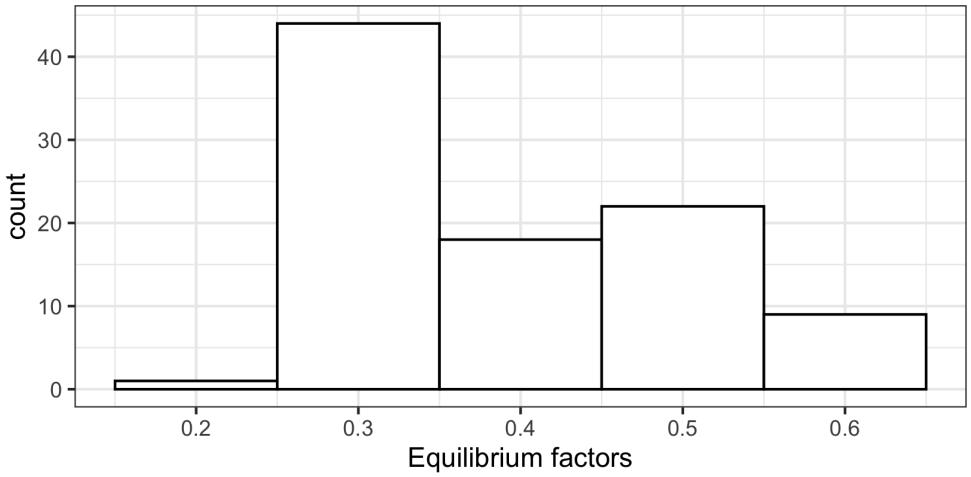


Figure 11: Equilibrium factors distribution

For the working time factors distribution the values from Küchenhoff et al. (2018, p.44) are used. They were weighted according to the number of years in which they occurred.

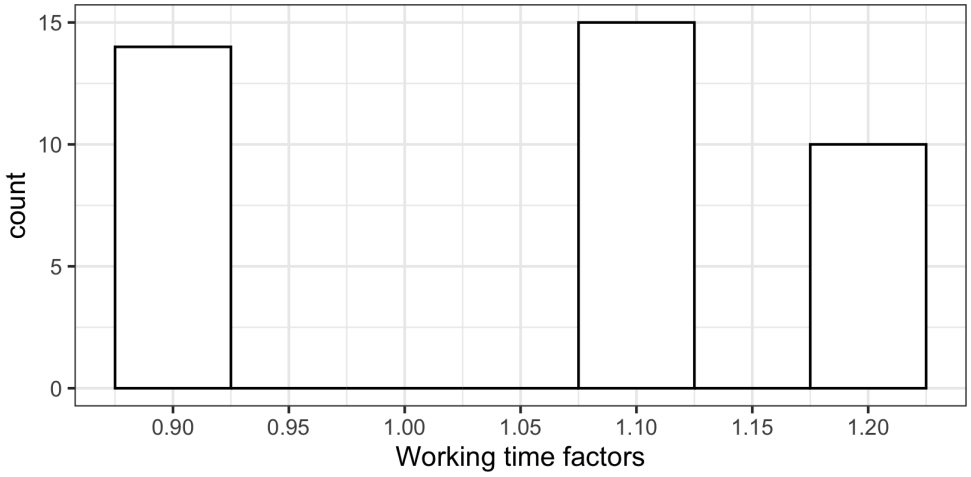


Figure 12: Working time factors distribution

### 4.1.3 Time of death generation

Modified forms of Hendry's function are used to generate the time of death for both data sets. In addition, it is ensured that each exposure period is longer than 0.001 to avoid problems when fitting the model.

### 4.1.4 Measurement error models

For all simulation studies, it is assumed that there is only one single influence variable, particularly the cumulative radon exposure in WLM/100. In the following,

the model calculated with $X^{cum}$ is referred to as the "true model" ($\mathcal{M}_0$)

$$\lambda_i(t) = \lambda_0(t)exp(X_i^{cum}(t)\beta) \tag{15}$$

($\mathcal{M}_0$), while the model calculated with $Z^{cum}$ is referred to as the "observed model"

$$\lambda_i(t) = \lambda_0(t)exp(Z_i^{cum}(t)\beta). \tag{16}$$

For the simulations, 0.05 was always used as the baseline hazard and 0.005 as the true beta.

**Simulation study 1: Approximation error due to rounding**

In the first simulation study, the impact of the approximation error (3.2.2) on the parameter estimation of the Cox model is considered. Three variants are examined here: the rounding to two, one and no decimal places.
Since the values in the fictive data set (data1) have only two decimal places, random noise is added to them in order to create true exposure values that have a higher precision. More on the exact implementation can be seen in the appendix (A.3).
The relationship between actual exposure $X_i$ of an individual $i$ and its observed exposure $Z_i$ can be expressed by the model

$$\mathcal{M}_1 : Z_i(t) = round(X_i(t), d) \tag{17}$$

where $d \in \{0, 1, 2\}$ is the decimal place to which $X_i$ is rounded.

**Simulation study 2: Assignment error**

Furthermore, the influence of the assignment error (3.2.2) is investigated in simulation study 2. The radon exposure values in data1 are considered as observed radon exposure, and in data2 $Z$ values are generated accordingly. Besides, various errors are generated. The true exposure is then calculated from the two quantities. Negative exposure values are set to zero.
Different variants are considered. On the one hand, models with unshared assignment errors, which can be seen in table 3, are examined. In this case, there is a separate error per year and worker, whereby in the homoscedastic case the errors are independent and identically distributed, and in the heteroscedastic case the variances differ depending on group membership of the worker. These groups can, for example, be occupational groups, or miner groups working in the same workplace. Both multiplicative and additive errors are considered, with the multiplicative errors being distributed log-normally and the additive errors

being distributed normally, as shown in table 3. In the heteroscedastic case, the individual variances are drawn from a normal distribution around the specified variance. That means in the multiplicative scenario from $N(\sigma^2, 0.05^2)$ and in the additive scenario from $N(\sigma^2, 10^2)$. If a negative variance is chosen, it is redrawn.

| homoscedastic/ heteroscedastic | multiplicative/ additive | model $\mathcal{M}$ and error $U$ |
|---|---|---|
| homoscedastic | multiplicative | $\mathcal{M}_2 : X_{ij}(t) = Z_j(t) \cdot U_i(t)$<br>$log(U_i(t)) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$ |
| | additive | $\mathcal{M}_3 : X_{ij}(t) = Z_j(t) + U_i(t)$<br>$U_i(t) \sim N(0, \sigma^2)$ |
| heteroscedastic | multiplicative | $\mathcal{M}_4 : X_{ij}(t) = Z_j(t) \cdot U_{ij}(t)$<br>$log(U_{ij}(t)) \sim N(-\frac{\sigma_j^2}{2}, \sigma_j^2)$ |
| | additive | $\mathcal{M}_5 : X_{ij}(t) = Z_j(t) + U_{ij}(t)$<br>$U_{ij}(t) \sim N(0, \sigma_j^2)$ |

Table 3: Assignment error models: unshared

In the models with shared assignment errors (table 4), a miner's error is constant over time. Analogous to the unshared models there are homoscedastic variants with the same variance for all errors and heteroscedastic variants with different ones per group, and also multiplicative and additive errors are considered.

| homoscedastic/ heteroscedastic | multiplicative/ additive | model $\mathcal{M}$ and error $U$ |
|---|---|---|
| homoscedastic | multiplicative | $\mathcal{M}_6 : X_{ij}(t) = Z_j(t) \cdot U_i$<br>$log(U_i) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$ |
| | additive | $\mathcal{M}_7 : X_{ij}(t) = Z_j(t) + U_i$<br>$U_i \sim N(0, \sigma^2)$ |
| heteroscedastic | multiplicative | $\mathcal{M}_8 : X_{ij}(t) = Z_j(t) \cdot U_{ij}$<br>$log(U_{ij}) \sim N(-\frac{\sigma_j^2}{2}, \sigma_j^2)$ |
| | additive | $\mathcal{M}_9 : X_{ij}(t) = Z_j(t) + U_{ij}$<br>$U_{ij} \sim N(0, \sigma_j^2)$ |

Table 4: Assignment error models: shared

Finally, unshared and shared errors are combined. There are two homoscedastic variations. One has a multiplicative unshared error and a multiplicative shared error and one is with an additive unshared error and an additive shared error. Also, there are heteroscedastic models, with either multiplicative or additive errors as well. The models can be seen in table 5.

| homoscedastic/ heteroscedastic | multiplicative/ additive | model $\mathcal{M}$ and error $U$ |
|---|---|---|
| homoscedastic | multiplicative | $\mathcal{M}_{10} : X_{ij}(t) = Z_j(t) \cdot U_i(t) \cdot U_i$<br>$log(U_i(t)) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$, $log(U_i) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$ |
| | additive | $\mathcal{M}_{11} : X_{ij}(t) = Z_j(t) + U_i(t) + U_i$<br>$U_i(t) \sim N(0, \sigma^2)$, $U_i \sim N(0, \sigma^2)$ |
| heteroscedastic | multiplicative | $\mathcal{M}_{12} : X_{ij}(t) = Z_j(t) \cdot U_{ij}(t) \cdot U_i$<br>$log(U_{ij}(t)) \sim N(-\frac{\sigma_j^2}{2}, \sigma_j^2)$, $log(U_i) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$ |
| | additive | $\mathcal{M}_{13} : X_{ij}(t) = Z_j(t) + U_i(t) + U_i$<br>$U_{ij}(t) \sim N(0, \sigma_j^2)$, $U_i \sim N(0, \sigma^2)$ |

Table 5: Assignment error models: unshared and shared

For the multiplicative errors the variances 0.77, 0.1 and 0.8 are used, and for the additive ones 190.18, 250.15 and 2721.73. More about the choice of error variances can be found in the appendix (A.2).

**Simulation study 3: Generalization error**

The third simulation study examined the generalization error. The fictive and generated exposure values are regarded as individual $X_i(t)$ values in both data sets. Based on this, a cluster version and a random version are simulated once each. First, the miners are randomly divided among 5 objects. It is assumed that each miner only works at one location.

For the cluster version, the workers within an object are split up into 5 locations. For this, the k-means method is used. The mean exposure values of the miners are used as the decisive values for clustering. The average of these values per object, location and year is also used as the true location value.

In the random version, the workers are randomly assigned to locations, just like with the objects. The true location value is calculated here by averaging the individual exposure values per object, location and year.

In both variants, the observed location value $Z_{jo}(t)$ results from the true location

value $X_{jo}(t)$ with a multiplied/added error:

$$\mathcal{M}_{14} : Z_{jo} = X_{jo} \cdot U_j(t), \tag{18}$$

$$\mathcal{M}_{15} : Z_{jo} = X_{jo} + U_j(t). \tag{19}$$

The error is different for each object, location and year. Like the assignment errors, the multiplicative generalization errors follow a log-normal distribution ($log(U_j(t)) \sim N(-\frac{\sigma^2}{2}, \sigma^2)$) and the additive ones a normal distribution ($U_j(t) \sim N(0, \sigma^2)$). If a negative observed location value results, it is set to zero. As error variances $\sigma^2$ for the multiplicative cases 0.0008, 0.01, 0.1 and 0.8 were used and for the additive cases 5.36, 64.68, 794.3066 and 5023.422. More about the choice of error variances can again be found in the appendix (A.2). The mean value of the observed location values $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} Z_{jo}(t)$ is calculated for each object and year. The cumulative true location values and the cumulative $\bar{Z}_o(t)$ values are used as covariates for the Cox models.

**Simulation study 4: Assignment and generalization error**

Subsequently, assignment and generalization errors are combined in the fourth simulation study. The procedure is analogous to simulation study 3. However, the true exposure values are not the true location values, but assignment errors are multiplied/added to them.

The assignment errors are again classified into unshared and shared errors. These are analogous to simulation study 2. However, only the homoscedastic case is considered here. In summary, the true exposure values are composed of the true location values and an error. The error can be unshared, i.e. different for each miner and year, or shared, i.e. there is only one error per worker. Table 6 summarises the resulting models.

| unshared/ shared | model $\mathcal{M}$ | $X$ and assignment error $U_i/U_i(t)$ | $Z$ and generalization error $U_j(t)$ |
|---|---|---|---|
| unshared | $\mathcal{M}_{16}$ | $X_{ijo}(t) = X_{jo}(t) \cdot U_i(t)$<br>$log(U_i(t)) \sim N(-\frac{\sigma_a^2}{2}, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} \cdot U_j(t)$<br>$log(U_j(t)) \sim N(-\frac{\sigma_g^2}{2}, \sigma_g^2)$ |
| | $\mathcal{M}_{17}$ | $X_{ijo}(t) = X_{jo}(t) + U_i(t)$<br>$U_i(t) \sim N(0, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} + U_j(t)$<br>$U_j(t) \sim N(0, \sigma_g^2)$ |
| shared | $\mathcal{M}_{18}$ | $X_{ijo}(t) = X_{jo}(t) \cdot U_i$<br>$log(U_i) \sim N(-\frac{\sigma_a^2}{2}, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} \cdot U_j(t)$<br>$log(U_j(t)) \sim N(-\frac{\sigma_g^2}{2}, \sigma_g^2)$ |
| | $\mathcal{M}_{19}$ | $X_{ijo}(t) = X_{jo}(t) + U_i$<br>$U_i \sim N(0, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} + U_j(t)$<br>$U_j(t) \sim N(0, \sigma_g^2)$ |
| both | $\mathcal{M}_{20}$ | $X_{ijo}(t) = X_{jo}(t) \cdot U_i(t) \cdot U_i$<br>$log(U_i(t)) \sim N(-\frac{\sigma_a^2}{2}, \sigma_a^2)$,<br>$U_i(t) \sim N(0, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} \cdot U_j(t)$<br>$log(U_j(t)) \sim N(-\frac{\sigma_g^2}{2}, \sigma_g^2)$ |
| | $\mathcal{M}_{21}$ | $X_{ijo}(t) = X_{jo}(t) + U_i(t) + U_i$<br>$U_i(t) \sim N(0, \sigma_a^2)$,<br>$U_i \sim N(0, \sigma_a^2)$ | $\bar{Z}_o(t) = \frac{1}{J}\sum_{j=1}^{J} X_{jo} + U_j(t)$<br>$U_j(t) \sim N(0, \sigma_g^2)$ |

Table 6: Assignment and generalization error models

The assignment error variances $\sigma_a^2$ are 0.077, 0.1 and 0.8 for multiplicative errors and 190.18, 250.15 and 2721.73 for additive errors. The generalization error variances $\sigma_g^2$ are 0.0008, 0.01, 0.1 and 0.8 for multiplicative errors and 5.36, 64.68, 794.31 and 5023.42 for additive errors.

**Simulation study 5: Estimation error**

Finally, in the last simulation study, parameter uncertainties in exposure estimations are investigated.

For the models with radon gas measurements, data2 was used. First, the miners are randomly assigned to an object and a shaft (in total there are five each). Then a mean radon gas value was drawn randomly from the distribution described in section 4.1.2 per object and year.

Then the different true factors, whose distributions are also defined there, are drawn randomly. It is assumed that each worker has only one activity over the entire work period, therefore each miner is assigned only one activity weighting factor $f(t, o, j)$ at random. Working time factors $w(t, o)$ and equilibrium factors $g(t, o)$ are sampled per worker and year because they can change over time.

For the estimates of the parameters $f(t,o,j)$, $w(t,o)$ and $g(t,o)$ their mean values are assumed with a multiplicative or additive error where the mean values are always calculated by averaging over the variables on which they depend. The errors are again log-normally or normally distributed, respectively. The following applies to multiplicative cases:

$$\hat{f}(t,o,j) = \bar{f}(t,o,j) \cdot U, \qquad log(U) \sim N(-\frac{2 \cdot \sigma_f^2}{2}, 2 \cdot \sigma_f^2) \tag{20}$$

$$\hat{w}(t,o) = \bar{w}(t,o) \cdot U, \qquad log(U) \sim N(-\frac{2 \cdot \sigma_w^2}{2}, 2 \cdot \sigma_w^2) \tag{21}$$

$$\hat{g}(t,o) = \bar{g}(t,o) \cdot U, \qquad log(U) \sim N(-\frac{2 \cdot \sigma_g^2}{2}, 2 \cdot \sigma_g^2). \tag{22}$$

Here $\sigma_i^2$ stands for the variance of the parameter $i \in \{f, w, g\}$.
The same applies in the additive case:

$$\hat{f}(t,o,j) = \bar{f}(t,o,j) + U, \qquad U \sim N(0, \sigma_f^2) \tag{23}$$

$$\hat{w}(t,o) = \bar{w}(t,o) + U, \qquad U \sim N(0, \sigma_w^2) \tag{24}$$

$$\hat{g}(t,o) = \bar{g}(t,o) + U, \qquad U \sim N(0, \sigma_g^2). \tag{25}$$

We thus obtain

$$X_i(t) = \bar{\bar{\bar{C}}}_{Rn}(t,o) \cdot 12 \cdot f(t,o,j) \cdot w(t,o) \cdot g(t,o) \tag{26}$$

$$Z_i(t) = \bar{\bar{\bar{C}}}_{Rn}(t,o) \cdot 12 \cdot \hat{f}(t,o,j) \cdot \hat{w}(t,o) \cdot \hat{g}(t,o). \tag{27}$$

An unshared error (per miner and year) and a shared error (per object and year) were used. For the fitting of the Cox models the cumulative exposure is used again.

For the models calculated with the radon progeny measurements, data1 is used. The mean radon progeny measurements are generated as described in section 4.1.2. The factors are sampled analogously to above, whereby the correction factor is used instead of the equilibrium factor. The other calculations also follow in the same way:

$$X_i(t) = \bar{\bar{\bar{C}}}_{RDP}(t,o) \cdot 12 \cdot f(t,o,j) \cdot w(t,o) \cdot c(o) \tag{28}$$

$$Z_i(t) = \bar{\bar{\bar{C}}}_{RDP}(t,o) \cdot 12 \cdot \hat{f}(t,o,j) \cdot \hat{w}(t,o) \cdot \hat{c}(o) \tag{29}$$

with

$$\hat{c}(o) = \bar{c}(o) \cdot U, \qquad log(U) \sim N(-\frac{2 \cdot \sigma_c^2}{2}, 2 \cdot \sigma_c^2) \tag{30}$$

or

$$\hat{c}(o) = \bar{c}(o) + U, \qquad U \sim N(0, \sigma_c^2). \tag{31}$$

The corresponding models resulting from this are shown in table 7.

| $\bar{\bar{C}}_{Rn}$/ $\bar{\bar{C}}_{RDP}$ | unshared / shared | model $\mathcal{M}$ | multiplicative/ additive | $Z$ |
|---|---|---|---|---|
| $\bar{\bar{C}}_{Rn}$ | unshared | $\mathcal{M}_{22}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot \hat{w} \cdot g$ |
| | | $\mathcal{M}_{23}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot \hat{w} \cdot g$ |
| | | $\mathcal{M}_{24}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot w \cdot \hat{g}$ |
| | | $\mathcal{M}_{25}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot w \cdot \hat{g}$ |
| | | $\mathcal{M}_{26}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot \hat{f} \cdot w \cdot g$ |
| | | $\mathcal{M}_{27}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot \hat{f} \cdot w \cdot g$ |
| | shared | $\mathcal{M}_{28}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot \hat{w} \cdot g$ |
| | | $\mathcal{M}_{29}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot \hat{w} \cdot g$ |
| | | $\mathcal{M}_{30}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot w \cdot \hat{g}$ |
| | | $\mathcal{M}_{31}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot f \cdot w \cdot \hat{g}$ |
| | | $\mathcal{M}_{32}$ | multiplicative | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot \hat{f} \cdot w \cdot g$ |
| | | $\mathcal{M}_{33}$ | additive | $Z = \bar{\bar{C}}_{Rn} \cdot 12 \cdot \hat{f} \cdot w \cdot g$ |
| $\bar{\bar{C}}_{RDP}$ | unshared | $\mathcal{M}_{34}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot \hat{w} \cdot c$ |
| | | $\mathcal{M}_{35}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot \hat{w} \cdot c$ |
| | | $\mathcal{M}_{36}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot w \cdot \hat{c}$ |
| | | $\mathcal{M}_{37}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot w \cdot \hat{c}$ |
| | | $\mathcal{M}_{38}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot \hat{f} \cdot w \cdot c$ |
| | | $\mathcal{M}_{39}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot \hat{f} \cdot w \cdot c$ |
| | shared | $\mathcal{M}_{40}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot \hat{w} \cdot c$ |
| | | $\mathcal{M}_{41}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot \hat{w} \cdot c$ |
| | | $\mathcal{M}_{42}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot w \cdot \hat{c}$ |
| | | $\mathcal{M}_{43}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot f \cdot w \cdot \hat{c}$ |
| | | $\mathcal{M}_{44}$ | multiplicative | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot \hat{f} \cdot w \cdot c$ |
| | | $\mathcal{M}_{45}$ | additive | $Z = \bar{\bar{C}}_{RDP} \cdot 12 \cdot \hat{f} \cdot w \cdot c$ |

Table 7: Estimation error models

The following variances are used:

- multiplicative:

$$\sigma_f^2 = 0.1450273 \tag{32}$$
$$\sigma_w^2 = 0.3475735 \tag{33}$$
$$\sigma_g^2 = 0.2271334 \tag{34}$$
$$\sigma_c^2 = 0.01133929 \tag{35}$$

- additive:

$$\sigma_f^2 = 0.07251366 \tag{36}$$
$$\sigma_w^2 = 0.01737868 \tag{37}$$
$$\sigma_g^2 = 0.01135667 \tag{38}$$
$$\sigma_c^2 = 0.005669643 \tag{39}$$

## 4.2 Results

For all models, the baseline hazard $\lambda_0(t) = 0.05$ and the true coefficient $\beta = 0.005$ per 100 WLM was assumed, and 500 replicates with 500 miners each were performed. In addition, each model variant was modeled once without censor and once with censor = 0.1. All values with more than 7 decimal places were rounded accordingly. The following values were calculated for all scenarios:

- $\hat{\beta}$: the mean of the coefficients $\beta$ of all repetitions
- RB: the relative bias $\frac{\hat{\beta}-0.005}{0.005}$
- CR: the coverage rate, i.e. the proportion of repetitions where the true $\beta = 0.005$ lies within the 95% confidence interval

For reasons of clarity and comprehensibility, not all result tables are shown in the main part of this thesis. More of them can be seen in the appendix (A.1).

### 4.2.1 Simulation study 1: Approximation error due to rounding

Table 8 shows the results of the approximation error simulation study. Neither the relative bias of the models with rounded exposure nor their coverage rate differs from the values of the true model.

| model | digits | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | 7 | none | 0.0046723 | -0.0655332 | 0.942 |
| | | 0.1 | 0.0047263 | -0.0547476 | 0.954 |
| $\mathcal{M}_1$ | 0 | none | 0.0046731 | -0.0653800 | 0.942 |
| | | 0.1 | 0.0047278 | -0.0544306 | 0.954 |
| | 1 | none | 0.0046724 | -0.0655292 | 0.942 |
| | | 0.1 | 0.0047263 | -0.0547358 | 0.954 |
| | 2 | none | 0.0046723 | -0.0655316 | 0.942 |
| | | 0.1 | 0.0047263 | -0.0547422 | 0.954 |

Table 8: Approximation error results (data1)

### 4.2.2 Simulation study 2: Assignment error

The tables in this chapter show the results of the simulation studies on the assignment error.

For the unshared assignment errors, there are no clear differences in relative bias and coverage rate between the models fitted with the real exposure and those fitted with the observed exposure. With shared errors, or unshared and shared ones combined, the difference between the relative biases is slightly greater.

It should be noted that the $\beta$ estimators of the models with true exposure values are different ones based on the error.
It is also noticeable that in the case of homoscedastic errors, the coverage rate of the observed model is always greater than or equal to the coverage rate of the real model for the unshared models.
In addition, in $\mathcal{M}_2$ and $\mathcal{M}_5$ it is striking that the absolute relative bias of the observed model is greater than the real model's one.

In the shared and combined unshared and shared models (tables 10 and 11), no trend can be discerned in this respect.

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_2$ | 0.077 | none | 0.0034036 | 0.0033596 | -0.3192831 | -0.3280822 | 0.956 | 0.962 |
| | | 0.1 | 0.0020990 | 0.0020903 | -0.5802058 | -0.5819395 | 0.942 | 0.942 |
| | 0.1 | none | 0.0038938 | 0.0037984 | -0.2212440 | -0.2403143 | 0.942 | 0.948 |
| | | 0.1 | 0.0035718 | 0.0034015 | -0.2856355 | -0.3197014 | 0.954 | 0.956 |
| | 0.8 | none | 0.0040106 | 0.0039493 | -0.1978859 | -0.2101341 | 0.950 | 0.950 |
| | | 0.1 | 0.0036885 | 0.0036325 | -0.2623069 | -0.2735089 | 0.948 | 0.950 |
| $\mathcal{M}_3$ | 190.18 | none | 0.0045988 | 0.0046372 | -0.0802384 | -0.0725699 | 0.928 | 0.928 |
| | | 0.1 | 0.0034777 | 0.0036675 | -0.3044596 | -0.2664955 | 0.938 | 0.948 |
| | 250.15 | none | 0.0037666 | 0.0037842 | -0.2466806 | -0.2431685 | 0.948 | 0.950 |
| | | 0.1 | 0.0030773 | 0.0034437 | -0.3845365 | -0.3112620 | 0.948 | 0.948 |
| | 2721.73 | none | 0.0045531 | 0.0045230 | -0.0893784 | -0.0953964 | 0.940 | 0.950 |
| | | 0.1 | 0.0057261 | 0.0056542 | 0.1452276 | 0.1308307 | 0.944 | 0.946 |
| $\mathcal{M}_4$ | 0.077 | none | 0.0043464 | 0.0043537 | -0.1307134 | -0.1292659 | 0.942 | 0.940 |
| | | 0.1 | 0.0022171 | 0.0024574 | -0.5565746 | -0.5085116 | 0.940 | 0.944 |
| | 0.1 | none | 0.0045117 | 0.0045652 | -0.0976657 | -0.0869576 | 0.952 | 0.944 |
| | | 0.1 | 0.0012224 | 0.0012933 | -0.7555131 | -0.7413402 | 0.954 | 0.956 |
| | 0.8 | none | 0.0049387 | 0.0049792 | -0.0122530 | -0.0041520 | 0.956 | 0.952 |
| | | 0.1 | 0.0040191 | 0.0037495 | -0.1961799 | -0.2501085 | 0.956 | 0.958 |
| $\mathcal{M}_5$ | 190.18 | none | 0.0049247 | 0.0048630 | -0.0150549 | -0.0273943 | 0.920 | 0.932 |
| | | 0.1 | 0.0023500 | 0.0023421 | -0.5299935 | -0.5315885 | 0.924 | 0.926 |
| | 250.15 | none | 0.0052749 | 0.0052897 | 0.0549773 | 0.0579459 | 0.952 | 0.946 |
| | | 0.1 | 0.0028985 | 0.0026836 | -0.4202986 | -0.4632830 | 0.958 | 0.960 |
| | 2721.73 | none | 0.0040064 | 0.0039552 | -0.1987213 | -0.2089603 | 0.962 | 0.960 |
| | | 0.1 | 0.0054721 | 0.0054919 | 0.0944155 | 0.0983704 | 0.928 | 0.934 |

Table 9: Assignment error results: unshared (data1)

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_6$ | 0.077 | none | 0.0049894 | 0.0049784 | -0.0021167 | -0.0043274 | 0.958 | 0.950 |
| | | 0.1 | 0.0021805 | 0.0022234 | -0.5638907 | -0.5553141 | 0.916 | 0.926 |
| | 0.1 | none | 0.0046460 | 0.0044934 | -0.0707939 | -0.1013165 | 0.950 | 0.942 |
| | | 0.1 | 0.0046722 | 0.0052391 | -0.0655694 | 0.04781206 | 0.946 | 0.938 |
| | 0.8 | none | 0.0042775 | 0.0041854 | -0.1445081 | -0.1629158 | 0.946 | 0.950 |
| | | 0.1 | 0.0058892 | 0.0064722 | 0.1778456 | 0.2944350 | 0.970 | 0.956 |
| $\mathcal{M}_7$ | 190.18 | none | 0.0046671 | 0.0046966 | -0.0665829 | -0.0606851 | 0.958 | 0.948 |
| | | 0.1 | 0.0024067 | 0.0020437 | -0.5186600 | -0.5912692 | 0.948 | 0.950 |
| | 250.15 | none | 0.0045234 | 0.0046539 | -0.0953291 | -0.0692232 | 0.932 | 0.936 |
| | | 0.1 | 0.0040899 | 0.0037226 | -0.1820159 | -0.2554857 | 0.950 | 0.952 |
| | 2721.73 | none | 0.0057763 | 0.0052649 | 0.1552548 | 0.0529734 | 0.958 | 0.966 |
| | | 0.1 | 0.0041710 | 0.0042124 | -0.1657996 | -0.1575216 | 0.944 | 0.936 |
| $\mathcal{M}_8$ | 0.077 | none | 0.0052276 | 0.0052260 | 0.0455157 | 0.0451992 | 0.944 | 0.942 |
| | | 0.1 | 0.0040279 | 0.0043570 | -0.1944235 | -0.1286039 | 0.942 | 0.938 |
| | 0.1 | none | 0.0052175 | 0.0054280 | 0.0434993 | 0.0856087 | 0.940 | 0.946 |
| | | 0.1 | 0.0033527 | 0.0027866 | -0.3294554 | -0.4426745 | 0.950 | 0.948 |
| | 0.8 | none | 0.0043202 | 0.0044479 | -0.1359661 | -0.1104131 | 0.940 | 0.934 |
| | | 0.1 | 0.0041154 | 0.0047901 | -0.1769205 | -0.0419873 | 0.944 | 0.944 |
| $\mathcal{M}_9$ | 190.18 | none | 0.0040715 | 0.0037831 | -0.1856966 | -0.2433900 | 0.966 | 0.966 |
| | | 0.1 | 0.0035863 | 0.0031374 | -0.2827482 | -0.3725115 | 0.940 | 0.942 |
| | 250.15 | none | 0.0049546 | 0.0045571 | -0.0090766 | -0.0885706 | 0.950 | 0.956 |
| | | 0.1 | 0.0051382 | 0.0051225 | 0.0276354 | 0.0245093 | 0.944 | 0.946 |
| | 2721.73 | none | 0.0052385 | 0.0052166 | 0.0477083 | 0.0433263 | 0.944 | 0.946 |
| | | 0.1 | 0.0047112 | 0.0048631 | -0.0577582 | -0.0273820 | 0.956 | 0.948 |

Table 10: Assignment error results: shared (data1)

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_{10}$ | 0.077 | none | 0.0047036 | 0.0047450 | -0.0592897 | -0.0509950 | 0.954 | 0.950 |
| | | 0.1 | 0.0036482 | 0.0037180 | -0.2703625 | -0.2564050 | 0.938 | 0.934 |
| | 0.1 | none | 0.0053561 | 0.0049794 | 0.0712135 | -0.0041265 | 0.932 | 0.934 |
| | | 0.1 | 0.0057420 | 0.0054551 | 0.1484087 | 0.0910274 | 0.940 | 0.938 |
| | 0.8 | none | 0.0041435 | 0.0039857 | -0.1712953 | -0.2028700 | 0.948 | 0.948 |
| | | 0.1 | 0.0040002 | 0.0035598 | -0.1999687 | -0.2880418 | 0.946 | 0.948 |
| $\mathcal{M}_{11}$ | 190.18 | none | 0.0039087 | 0.0039144 | -0.2182676 | -0.2171181 | 0.952 | 0.952 |
| | | 0.1 | 0.0027383 | 0.0026999 | -0.4523458 | -0.4600252 | 0.932 | 0.928 |
| | 250.15 | none | 0.0045167 | 0.0042615 | -0.0966531 | -0.1476901 | 0.944 | 0.950 |
| | | 0.1 | 0.0038652 | 0.0042699 | -0.2269585 | -0.1460297 | 0.946 | 0.964 |
| | 2721.73 | none | 0.0044261 | 0.0045122 | -0.1147826 | -0.0975589 | 0.916 | 0.912 |
| | | 0.1 | 0.0050385 | 0.0051066 | 0.0077025 | 0.0213189 | 0.942 | 0.940 |
| $\mathcal{M}_{12}$ | 0.077 | none | 0.0048253 | 0.0047265 | -0.0349367 | -0.0547061 | 0.936 | 0.930 |
| | | 0.1 | 0.0027375 | 0.0031795 | -0.4525050 | -0.3641035 | 0.944 | 0.938 |
| | 0.1 | none | 0.0051854 | 0.0050348 | 0.0370805 | 0.0069619 | 0.952 | 0.958 |
| | | 0.1 | 0.0042836 | 0.0038303 | -0.1432899 | -0.2339399 | 0.954 | 0.954 |
| | 0.8 | none | 0.0051287 | 0.0050557 | 0.0257377 | 0.0111361 | 0.936 | 0.946 |
| | | 0.1 | 0.0044221 | 0.0041122 | -0.1155897 | -0.1775576 | 0.946 | 0.954 |
| $\mathcal{M}_{13}$ | 190.18 | none | 0.0046459 | 0.0044959 | -0.0708233 | -0.1008246 | 0.968 | 0.962 |
| | | 0.1 | 0.0046693 | 0.0040806 | -0.0661306 | -0.1838790 | 0.934 | 0.946 |
| | 250.15 | none | 0.0050369 | 0.0049887 | 0.0073712 | -0.0022630 | 0.940 | 0.948 |
| | | 0.1 | 0.0041042 | 0.0033937 | -0.1791609 | -0.3212609 | 0.940 | 0.944 |
| | 2721.73 | none | 0.0038449 | 0.0038187 | -0.2310300 | -0.2362678 | 0.950 | 0.948 |
| | | 0.1 | 0.0036069 | 0.0035879 | -0.2786234 | -0.2824140 | 0.946 | 0.942 |

Table 11: Assignment error results: unshared and shared (data1)

### 4.2.3 Simulation study 3: Generalization error

The results of the generalization error simulation study show larger differences. A clear trend can be seen in the cluster version with additive measurement error (table 12). The greater the error variance is, the greater the absolute relative bias. The coverage rate, on the other hand, varies only slightly.
In the case of multiplicative errors, on the other hand, it is the other way round. The greater the variance of the errors, the smaller the absolute relative bias generally becomes.

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0049269 | -0.0146116 | 0.956 |
| | | 0.1 | 0.0038457 | -0.2308508 | 0.958 |
| $\mathcal{M}_{14}$ | 0.0008 | none | 0.0044853 | -0.1029392 | 0.956 |
| | | 0.1 | 0.0039764 | -0.2047188 | 0.956 |
| | 0.01 | none | 0.0045158 | -0.0968316 | 0.956 |
| | | 0.1 | 0.0038100 | -0.2380082 | 0.954 |
| | 0.1 | none | 0.0046880 | -0.0623994 | 0.956 |
| | | 0.1 | 0.0040591 | -0.1881788 | 0.958 |
| | 0.8 | none | 0.0049631 | -0.0073810 | 0.960 |
| | | 0.1 | 0.0044569 | -0.1086138 | 0.952 |
| $\mathcal{M}_{15}$ | 5.36 | none | 0.0044639 | -0.1072300 | 0.954 |
| | | 0.1 | 0.0040107 | -0.1978698 | 0.954 |
| | 64.68 | none | 0.0043958 | -0.1208428 | 0.954 |
| | | 0.1 | 0.0037862 | -0.2427616 | 0.962 |
| | 794.31 | none | 0.0037251 | -0.2549840 | 0.960 |
| | | 0.1 | 0.0029591 | -0.4081816 | 0.950 |
| | 5023.42 | none | 0.0021277 | -0.5744650 | 0.954 |
| | | 0.1 | 0.0020078 | -0.5984460 | 0.964 |

Table 12: Generalization error results: cluster (data1)

The results of the random version, that are shown in table 13, are not so clear. Only in the additive model without censor, similar structures can be recognized as in the cluster version.

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0029877 | -0.4024626 | 0.952 |
| | | 0.1 | 0.0048828 | -0.0234402 | 0.954 |
| $\mathcal{M}_{14}$ | 0.0008 | none | 0.0029543 | -0.4091500 | 0.956 |
| | | 0.1 | 0.0049302 | -0.0139581 | 0.954 |
| | 0.01 | none | 0.0029151 | -0.4169712 | 0.956 |
| | | 0.1 | 0.0048896 | -0.0220853 | 0.954 |
| | 0.1 | none | 0.0028987 | -0.4202520 | 0.950 |
| | | 0.1 | 0.0049306 | -0.0138810 | 0.958 |
| | 0.8 | none | 0.0030379 | -0.3924128 | 0.960 |
| | | 0.1 | 0.0040392 | -0.1921579 | 0.952 |
| $\mathcal{M}_{15}$ | 5.36 | none | 0.0029551 | -0.4089772 | 0.954 |
| | | 0.1 | 0.0049259 | -0.0148289 | 0.954 |
| | 64.68 | none | 0.0029440 | -0.4111924 | 0.950 |
| | | 0.1 | 0.0050440 | 0.0088086 | 0.954 |
| | 794.31 | none | 0.0024498 | -0.5100478 | 0.946 |
| | | 0.1 | 0.0050743 | 0.0148655 | 0.964 |
| | 5023.42 | none | 0.0012974 | -0.7405126 | 0.944 |
| | | 0.1 | 0.0032896 | -0.3420797 | 0.954 |

Table 13: Generalization error results: random (data1)

### 4.2.4 Simulation study 4: Assignment and generalization error

The models with the assignment and generalization errors do not show as strong tendencies as those with generalization errors alone.

However, for $\mathcal{M}_{16}$, the absolute relative bias of the observed model is greater than the one of the X-model. With a small assignment error variance (0.077), the coverage rate of the true model is always higher or equal to the one of the observed model.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 0.077 | 0.0008 | none | 0.0044535 | 0.0055587 | -0.1092964 | 0.1117427 | 0.952 | 0.936 |
| | | 0.1 | 0.0045257 | 0.0035674 | -0.0948580 | -0.2865196 | 0.958 | 0.958 |
| | 0.01 | none | 0.0044535 | 0.0055564 | -0.1092964 | 0.1112778 | 0.952 | 0.938 |
| | | 0.1 | 0.0045257 | 0.0035496 | -0.0948580 | -0.2900834 | 0.958 | 0.954 |
| | 0.1 | none | 0.0044535 | 0.0055311 | -0.1092964 | 0.1062112 | 0.952 | 0.938 |
| | | 0.1 | 0.0045257 | 0.0033169 | -0.0948580 | -0.3366282 | 0.958 | 0.956 |
| | 0.8 | none | 0.0044535 | 0.0053361 | -0.1092964 | 0.0672127 | 0.952 | 0.938 |
| | | 0.1 | 0.0045257 | 0.0040965 | -0.0948580 | -0.1807032 | 0.958 | 0.956 |
| 0.1 | 0.0008 | none | 0.0051070 | 0.0046668 | 0.0213963 | -0.0666342 | 0.940 | 0.962 |
| | | 0.1 | 0.0044067 | 0.0035437 | -0.1186518 | -0.2912680 | 0.950 | 0.950 |
| | 0.01 | none | 0.0051070 | 0.0047024 | 0.0213963 | -0.0595115 | 0.940 | 0.960 |
| | | 0.1 | 0.0044067 | 0.0035354 | -0.1186518 | -0.2929152 | 0.950 | 0.948 |
| | 0.1 | none | 0.0051070 | 0.0046897 | 0.0213963 | -0.0620571 | 0.940 | 0.960 |
| | | 0.1 | 0.0044067 | 0.0033091 | -0.1186518 | -0.3381768 | 0.950 | 0.952 |
| | 0.8 | none | 0.0051070 | 0.0042729 | 0.0213963 | -0.1454253 | 0.940 | 0.948 |
| | | 0.1 | 0.0044067 | 0.0042061 | -0.1186518 | -0.1587771 | 0.950 | 0.950 |
| 0.8 | 0.0008 | none | 0.0054574 | 0.0036193 | 0.0914807 | -0.2761368 | 0.952 | 0.940 |
| | | 0.1 | 0.0021473 | 0.0021708 | -0.5705420 | -0.5658334 | 0.956 | 0.960 |
| | 0.01 | none | 0.0054574 | 0.0036498 | 0.0914807 | -0.2700453 | 0.952 | 0.942 |
| | | 0.1 | 0.0021473 | 0.0021830 | -0.5705420 | -0.5634077 | 0.956 | 0.960 |
| | 0.1 | none | 0.0054574 | 0.0035063 | 0.0914807 | -0.2987466 | 0.952 | 0.944 |
| | | 0.1 | 0.0021473 | 0.0020095 | -0.5705420 | -0.5981082 | 0.956 | 0.964 |
| | 0.8 | none | 0.0054574 | 0.0035054 | 0.0914807 | -0.2989278 | 0.952 | 0.934 |
| | | 0.1 | 0.0021473 | 0.0032629 | -0.5705420 | -0.3474276 | 0.956 | 0.956 |

Table 14: Assignment and generalization error results: $\mathcal{M}_{16}$, cluster, unshared, multiplicative (data1)

In Model $\mathcal{M}_{17}$ with small error variances, the absolute relative bias of the true model is often larger than that of the observed model. The larger the variance, however, the more often it is reversed. This applies to the assignment error variance as well as to the generalization error variance.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 190.18 | 5.36 | none | 0.0045779 | 0.0046825 | -0.0844256 | -0.0635010 | 0.950 | 0.936 |
| | | 0.1 | 0.0037852 | 0.0048737 | -0.2429692 | -0.0252618 | 0.950 | 0.964 |
| | 64.68 | none | 0.0045779 | 0.0046040 | -0.0844256 | -0.0791978 | 0.950 | 0.934 |
| | | 0.1 | 0.0037852 | 0.0050615 | -0.2429692 | 0.0123008 | 0.950 | 0.962 |
| | 794.31 | none | 0.0045779 | 0.0038845 | -0.0844256 | -0.2231003 | 0.950 | 0.930 |
| | | 0.1 | 0.0037852 | 0.0038013 | -0.2429692 | -0.2397490 | 0.950 | 0.960 |
| | 5023.42 | none | 0.0045779 | 0.0036924 | -0.0844256 | -0.2615273 | 0.950 | 0.938 |
| | | 0.1 | 0.0037852 | 0.0036095 | -0.2429692 | -0.2780917 | 0.950 | 0.948 |
| 250.15 | 5.36 | none | 0.0046422 | 0.0050707 | -0.0715616 | 0.0141420 | 0.948 | 0.944 |
| | | 0.1 | 0.0036666 | 0.0023575 | -0.2666881 | -0.5284993 | 0.944 | 0.956 |
| | 64.68 | none | 0.0046422 | 0.0052093 | -0.0715616 | 0.0418626 | 0.948 | 0.948 |
| | | 0.1 | 0.0036666 | 0.0024883 | -0.2666881 | -0.5023489 | 0.944 | 0.958 |
| | 794.31 | none | 0.0046422 | 0.0041644 | -0.0715616 | -0.1671215 | 0.948 | 0.938 |
| | | 0.1 | 0.0036666 | 0.0022302 | -0.2666881 | -0.5539613 | 0.944 | 0.964 |
| | 5023.42 | none | 0.0046422 | 0.0029686 | -0.0715616 | -0.4062734 | 0.948 | 0.946 |
| | | 0.1 | 0.0036666 | 0.0024170 | -0.2666881 | -0.5165994 | 0.944 | 0.948 |
| 2721.73 | 5.36 | none | 0.0050680 | 0.0064158 | 0.01359421 | 0.2831527 | 0.968 | 0.946 |
| | | 0.1 | 0.0018172 | 0.0027347 | -0.6365516 | -0.4530567 | 0.962 | 0.962 |
| | 64.68 | none | 0.0050680 | 0.0063895 | 0.01359421 | 0.2778982 | 0.968 | 0.944 |
| | | 0.1 | 0.0018172 | 0.0026656 | -0.6365516 | -0.4668732 | 0.962 | 0.964 |
| | 794.31 | none | 0.0050680 | 0.0056298 | 0.01359421 | 0.1259555 | 0.968 | 0.944 |
| | | 0.1 | 0.0018172 | -0.0012316 | -0.6365516 | -1.2463100 | 0.962 | 0.952 |
| | 5023.42 | none | 0.0050680 | 0.0046381 | 0.01359421 | -0.0723856 | 0.968 | 0.952 |
| | | 0.1 | 0.0018172 | 0.0009103 | -0.6365516 | -0.8179308 | 0.962 | 0.942 |

Table 15: Assignment and generalization error results: $\mathcal{M}_{17}$, cluster, unshared, additive (data1)

For $\mathcal{M}_{18}$ applies:
With an assignment error variance of 0.007, the coverage rate of the true model is always greater than or equal to the coverage rate of the observed model. With an assignment error variance of 0.1, $|RBX| > |RBZ|$ applies, with a variance of 0.8, the opposite is the case.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 0.077 | 0.0008 | none | 0.0056380 | 0.0050935 | 0.1275986 | 0.0186936 | 0.960 | 0.950 |
| | | 0.1 | 0.0030759 | 0.0029798 | -0.3848298 | -0.4040356 | 0.964 | 0.964 |
| | 0.01 | none | 0.0056380 | 0.0051166 | 0.1275986 | 0.0233233 | 0.960 | 0.946 |
| | | 0.1 | 0.0030759 | 0.0029794 | -0.3848298 | -0.4041226 | 0.964 | 0.962 |
| | 0.1 | none | 0.0056380 | 0.0051014 | 0.1275986 | 0.0202719 | 0.960 | 0.948 |
| | | 0.1 | 0.0030759 | 0.0028010 | -0.3848298 | -0.4398038 | 0.964 | 0.958 |
| | 0.8 | none | 0.0056380 | 0.0047759 | 0.1275986 | -0.0448150 | 0.960 | 0.946 |
| | | 0.1 | 0.0030759 | 0.0038678 | -0.3848298 | -0.2264459 | 0.964 | 0.960 |
| 0.1 | 0.0008 | none | 0.0053031 | 0.0052674 | 0.0606298 | 0.0534863 | 0.946 | 0.944 |
| | | 0.1 | 0.0038682 | 0.0056740 | -0.2263658 | 0.1348055 | 0.942 | 0.962 |
| | 0.01 | none | 0.0053031 | 0.0052634 | 0.0606298 | 0.0526717 | 0.946 | 0.944 |
| | | 0.1 | 0.0038682 | 0.0056579 | -0.2263658 | 0.1315706 | 0.942 | 0.964 |
| | 0.1 | none | 0.0053031 | 0.0051880 | 0.0606298 | 0.0376071 | 0.946 | 0.946 |
| | | 0.1 | 0.0038682 | 0.0054655 | -0.2263658 | 0.0930934 | 0.942 | 0.964 |
| | 0.8 | none | 0.0053031 | 0.0052772 | 0.0606298 | 0.0554404 | 0.946 | 0.946 |
| | | 0.1 | 0.0038682 | 0.0058142 | -0.2263658 | 0.1628371 | 0.942 | 0.962 |
| 0.8 | 0.0008 | none | 0.0059330 | 0.0034621 | 0.1866098 | -0.3075813 | 0.948 | 0.952 |
| | | 0.1 | 0.0035754 | 0.0010717 | -0.2849212 | -0.7856612 | 0.952 | 0.938 |
| | 0.01 | none | 0.0059330 | 0.0035083 | 0.1866098 | -0.2983421 | 0.948 | 0.954 |
| | | 0.1 | 0.0035754 | 0.0010167 | -0.2849212 | -0.7966622 | 0.952 | 0.936 |
| | 0.1 | none | 0.0059330 | 0.0034639 | 0.1866098 | -0.3072168 | 0.948 | 0.958 |
| | | 0.1 | 0.0035754 | 0.0011374 | -0.2849212 | -0.7725214 | 0.952 | 0.934 |
| | 0.8 | none | 0.0059330 | 0.0032993 | 0.1866098 | -0.3401351 | 0.948 | 0.950 |
| | | 0.1 | 0.0035754 | 0.0009769 | -0.2849212 | -0.8046237 | 0.952 | 0.954 |

Table 16: Assignment and generalization error results: $\mathcal{M}_{18}$, cluster, shared, multiplicative (data1)

For $\mathcal{M}_{19}$ with $\sigma^2_{as} = 2721.73$, the absolute relative bias of the true model is usually smaller than that of the observed model. The coverage rate ratio usually remains the same.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 190.18 | 5.36 | none | 0.0036208 | 0.0037268 | -0.2758414 | -0.2546304 | 0.940 | 0.948 |
| | | 0.1 | 0.0053519 | 0.0052025 | 0.0703734 | 0.0404922 | 0.950 | 0.938 |
| | 64.68 | none | 0.0036208 | 0.0038014 | -0.2758414 | -0.2397106 | 0.940 | 0.952 |
| | | 0.1 | 0.0053519 | 0.0049276 | 0.0703734 | -0.0144714 | 0.950 | 0.940 |
| | 794.31 | none | 0.0036208 | 0.0024958 | -0.2758414 | -0.5008492 | 0.940 | 0.948 |
| | | 0.1 | 0.0053519 | 0.0017609 | 0.0703734 | -0.6478148 | 0.950 | 0.948 |
| | 5023.42 | none | 0.0036208 | 0.0024299 | -0.2758414 | -0.5140257 | 0.940 | 0.940 |
| | | 0.1 | 0.0053519 | 0.0029793 | 0.0703734 | -0.4041440 | 0.950 | 0.928 |
| 250.15 | 5.36 | none | 0.0041413 | 0.0054441 | -0.1717482 | 0.0888114 | 0.948 | 0.932 |
| | | 0.1 | 0.0026216 | 0.0020812 | -0.4756781 | -0.5837543 | 0.952 | 0.960 |
| | 64.68 | none | 0.0041413 | 0.0053372 | -0.1717482 | 0.0674313 | 0.948 | 0.942 |
| | | 0.1 | 0.0026216 | 0.0021971 | -0.4756781 | -0.5605707 | 0.952 | 0.954 |
| | 794.31 | none | 0.0041413 | 0.0046743 | -0.1717482 | -0.0651388 | 0.948 | 0.942 |
| | | 0.1 | 0.0026216 | 0.0015537 | -0.4756781 | -0.6892584 | 0.952 | 0.946 |
| | 5023.42 | none | 0.0041413 | 0.0028397 | -0.1717482 | -0.4320544 | 0.948 | 0.946 |
| | | 0.1 | 0.0026216 | 0.0055590 | -0.4756781 | 0.1117924 | 0.952 | 0.942 |
| 2721.73 | 5.36 | none | 0.0034990 | 0.0026505 | -0.3002033 | -0.4699072 | 0.922 | 0.945 |
| | | 0.1 | 0.0025534 | 0.0018187 | -0.4893127 | -0.6362508 | 0.974 | 0.944 |
| | 64.68 | none | 0.0034990 | 0.0025952 | -0.3002033 | -0.4809623 | 0.922 | 0.954 |
| | | 0.1 | 0.0025534 | 0.0020032 | -0.4893127 | -0.5993680 | 0.974 | 0.944 |
| | 794.31 | none | 0.0034990 | 0.0019583 | -0.3002033 | -0.6083408 | 0.922 | 0.958 |
| | | 0.1 | 0.0025534 | 0.0009963 | -0.4893127 | -0.8007429 | 0.974 | 0.948 |
| | 5023.42 | none | 0.0034990 | -0.0001572 | -0.3002033 | -1.0314350 | 0.922 | 0.944 |
| | | 0.1 | 0.0025534 | 0.0042512 | -0.4893127 | -0.1497583 | 0.974 | 0.942 |

Table 17: Assignment and generalization error results: $\mathcal{M}_{19}$, cluster, shared, additive (data1)

The coverage rate ratio between the observed and the true model usually remains the same for $\mathcal{M}_{20}$. With small and medium assignment error variance (0.007 and 0.1), the absolute relative bias is always smaller than that of the observed model.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 0.077 | 0.0008 | none | 0.0056650 | 0.0041967 | 0.1330007 | -0.1606619 | 0.952 | 0.962 |
| | | 0.1 | 0.0030957 | 0.0026289 | -0.3808641 | -0.4742185 | 0.962 | 0.954 |
| | 0.01 | none | 0.0056650 | 0.0041967 | 0.1330007 | -0.1606553 | 0.952 | 0.960 |
| | | 0.1 | 0.0030957 | 0.0026047 | -0.3808641 | -0.4790629 | 0.962 | 0.954 |
| | 0.1 | none | 0.0056650 | 0.0042058 | 0.1330007 | -0.1588368 | 0.952 | 0.960 |
| | | 0.1 | 0.0030957 | 0.0024743 | -0.3808641 | -0.5051418 | 0.962 | 0.954 |
| | 0.8 | none | 0.0056650 | 0.0037534 | 0.1330007 | -0.2493109 | 0.952 | 0.956 |
| | | 0.1 | 0.0030957 | 0.0029952 | -0.3808641 | -0.4009512 | 0.962 | 0.952 |
| 0.1 | 0.0008 | none | 0.0047798 | 0.0039767 | -0.0440500 | -0.2046630 | 0.964 | 0.950 |
| | | 0.1 | 0.0041344 | 0.0065328 | -0.1731243 | 0.3065565 | 0.948 | 0.962 |
| | 0.01 | none | 0.0047798 | 0.0039992 | -0.0440500 | -0.2001695 | 0.964 | 0.948 |
| | | 0.1 | 0.0041344 | 0.0064900 | -0.1731243 | 0.2980077 | 0.948 | 0.964 |
| | 0.1 | none | 0.0047798 | 0.0040088 | -0.0440500 | -0.1982445 | 0.964 | 0.942 |
| | | 0.1 | 0.0041344 | 0.0062275 | -0.1731243 | 0.2454979 | 0.948 | 0.964 |
| | 0.8 | none | 0.0047798 | 0.0038494 | -0.0440500 | -0.2301166 | 0.964 | 0.950 |
| | | 0.1 | 0.0041344 | 0.0068166 | -0.1731243 | 0.3633169 | 0.948 | 0.960 |
| 0.8 | 0.0008 | none | 0.0073132 | 0.0044020 | 0.4626407 | -0.119602 | 0.948 | 0.940 |
| | | 0.1 | 0.0015338 | -0.0005810 | -0.6932466 | -1.116203 | 0.948 | 0.948 |
| | 0.01 | none | 0.0073132 | 0.0044390 | 0.4626407 | -0.112199 | 0.948 | 0.940 |
| | | 0.1 | 0.0015338 | -0.0004877 | -0.6932466 | -1.097539 | 0.948 | 0.946 |
| | 0.1 | none | 0.0073132 | 0.0045121 | 0.4626407 | -0.0975827 | 0.948 | 0.938 |
| | | 0.1 | 0.0015338 | -0.0006259 | -0.6932466 | -1.1251700 | 0.948 | 0.942 |
| | 0.8 | none | 0.0073132 | 0.0037065 | 0.4626407 | -0.2586939 | 0.948 | 0.944 |
| | | 0.1 | 0.0015338 | -0.0002043 | -0.6932466 | -1.0408700 | 0.948 | 0.944 |

Table 18: Assignment and generalization error results: $\mathcal{M}_{20}$, cluster, unshared and shared, multiplicative (data1)

In the clustered case with additive unshared and shared assignment error ($\mathcal{M}_{21}$), the true coverage rate for small error variants ($\sigma^2_{as}$ and $\sigma^2_{ge}$) is smaller than the observed one, for larger variances, it is mostly the other way round. For simulations without censor applies that the greater the generalization error variance, the greater the absolute relative bias.

| $\sigma^2_{as}$ | $\sigma^2_{ge}$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| 190.18 | 5.36 | none | 0.0037988 | 0.0031062 | -0.2402310 | 0.3787521 | 0.938 | 0.944 |
| | | 0.1 | 0.0026081 | 0.0067170 | -0.4783874 | 0.3433987 | 0.956 | 0.960 |
| | 64.68 | none | 0.0037988 | 0.0029891 | -0.2402310 | 0.4021881 | 0.938 | 0.944 |
| | | 0.1 | 0.0026081 | 0.0070254 | -0.4783874 | 0.4050777 | 0.956 | 0.964 |
| | 794.31 | none | 0.0037988 | 0.0019254 | -0.2402310 | 0.6149202 | 0.938 | 0.940 |
| | | 0.1 | 0.0026081 | 0.0063933 | -0.4783874 | 0.2786661 | 0.956 | 0.952 |
| | 5023.42 | none | 0.0037988 | 0.0016766 | -0.2402310 | 0.6646897 | 0.938 | 0.950 |
| | | 0.1 | 0.0026081 | 0.0057219 | -0.4783874 | 0.1443719 | 0.956 | 0.942 |
| 250.15 | 5.36 | none | 0.0041347 | 0.0042194 | -0.1730606 | -0.1561194 | 0.956 | 0.952 |
| | | 0.1 | 0.0029269 | 0.0039594 | -0.4146180 | -0.2081114 | 0.962 | 0.950 |
| | 64.68 | none | 0.0041347 | 0.0041111 | -0.1730606 | -0.1777765 | 0.956 | 0.952 |
| | | 0.1 | 0.0029269 | 0.0044777 | -0.4146180 | -0.1044631 | 0.962 | 0.946 |
| | 794.31 | none | 0.0041347 | 0.0033219 | -0.1730606 | -0.3356244 | 0.956 | 0.952 |
| | | 0.1 | 0.0029269 | 0.0022719 | -0.4146180 | -0.5456138 | 0.962 | 0.952 |
| | 5023.42 | none | 0.0041347 | 0.0020746 | -0.1730606 | -0.5850873 | 0.956 | 0.954 |
| | | 0.1 | 0.0029269 | 0.0049078 | -0.4146180 | -0.0184303 | 0.962 | 0.938 |
| 2721.73 | 5.36 | none | 0.0032631 | 0.0049284 | -0.3473880 | -0.0143250 | 0.940 | 0.938 |
| | | 0.1 | 0.0025281 | 0.0034547 | -0.4943821 | -0.3090688 | 0.954 | 0.944 |
| | 64.68 | none | 0.0032631 | 0.0049474 | -0.3473880 | -0.0105248 | 0.940 | 0.944 |
| | | 0.1 | 0.0025281 | 0.0035883 | -0.4943821 | -0.2823405 | 0.954 | 0.948 |
| | 794.31 | none | 0.0032631 | 0.0043865 | -0.3473880 | -0.1227013 | 0.940 | 0.954 |
| | | 0.1 | 0.0025281 | 0.0018964 | -0.4943821 | -0.6207132 | 0.954 | 0.952 |
| | 5023.42 | none | 0.0032631 | 0.0015328 | -0.3473880 | -0.6934423 | 0.940 | 0.930 |
| | | 0.1 | 0.0025281 | 0.0028143 | -0.4943821 | -0.4371378 | 0.954 | 0.936 |

Table 19: Assignment and generalization error results: $\mathcal{M}_{21}$, cluster, unshared and shared, additive (data1)

### 4.2.5 Simulation study 5: Estimation error

All estimation error simulation studies with the estimated activity factor ($\mathcal{M}_{26}, \mathcal{M}_{27}, \mathcal{M}_{32}, \mathcal{M}_{33}, \mathcal{M}_{38}, \mathcal{M}_{39}, \mathcal{M}_{44}$ and $\mathcal{M}_{45}$) show that the relative bias of the observed model deviates strongly from the true model. The other parameters do not show such clear differences.

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0076335 | 0.5266993 | 0.950 |
| | | 0.1 | 0.0058780 | 0.1755911 | 0.952 |
| $\mathcal{M}_{22}$ | 0.3475735 | none | 0.0075059 | 0.5011745 | 0.952 |
| | | 0.1 | 0.0077056 | 0.5411178 | 0.950 |
| $\mathcal{M}_{23}$ | 0.01737868 | none | 0.0075075 | 0.5015095 | 0.944 |
| | | 0.1 | 0.0063024 | 0.2604707 | 0.948 |
| $\mathcal{M}_{24}$ | 0.2271334 | none | 0.0078883 | 0.5776673 | 0.952 |
| | | 0.1 | 0.0056452 | 0.1290333 | 0.956 |
| $\mathcal{M}_{25}$ | 0.01135667 | none | 0.0077537 | 0.5507309 | 0.946 |
| | | 0.1 | 0.0061886 | 0.2377110 | 0.956 |
| $\mathcal{M}_{26}$ | 0.1450273 | none | 0.0046247 | -0.0750638 | 0.962 |
| | | 0.1 | 0.0088339 | 0.7667879 | 0.944 |
| $\mathcal{M}_{27}$ | 0.07251366 | none | 0.0045960 | -0.0808036 | 0.950 |
| | | 0.1 | 0.0097736 | 0.9547235 | 0.952 |

Table 20: Estimation error results: unshared (data2)

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0076335 | 0.5266993 | 0.950 |
| | | 0.1 | 0.0058780 | 0.1755911 | 0.952 |
| $\mathcal{M}_{28}$ | 0.3475735 | none | 0.0076205 | 0.5241072 | 0.952 |
| | | 0.1 | 0.0067819 | 0.3563845 | 0.948 |
| $\mathcal{M}_{29}$ | 0.01737868 | none | 0.0074658 | 0.4931644 | 0.948 |
| | | 0.1 | 0.0070790 | 0.4157939 | 0.954 |
| $\mathcal{M}_{30}$ | 0.2271334 | none | 0.0082244 | 0.6448780 | 0.950 |
| | | 0.1 | 0.0064179 | 0.2835876 | 0.956 |
| $\mathcal{M}_{31}$ | 0.01135667 | none | 0.0080973 | 0.6194531 | 0.950 |
| | | 0.1 | 0.0082427 | 0.6485322 | 0.956 |
| $\mathcal{M}_{32}$ | 0.1450273 | none | 0.007842927 | 0.5685853 | 0.952 |
| | | 0.1 | 0.01189386 | 1.3787720 | 0.944 |
| $\mathcal{M}_{33}$ | 0.07251366 | none | 0.004903754 | -0.0192492 | 0.946 |
| | | 0.1 | 0.005148285 | 0.0296569 | 0.944 |

Table 21: Estimation error results: shared (data2)

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0052460 | 0.0491988 | 0.942 |
| | | 0.1 | 0.0047003 | -0.0599446 | 0.952 |
| $\mathcal{M}_{34}$ | 0.3475735 | none | 0.0052606 | 0.0521222 | 0.940 |
| | | 0.1 | 0.0045977 | -0.0804571 | 0.952 |
| $\mathcal{M}_{35}$ | 0.01737868 | none | 0.0052816 | 0.0563161 | 0.942 |
| | | 0.1 | 0.0045987 | -0.0802600 | 0.950 |
| $\mathcal{M}_{36}$ | 0.01133929 | none | 0.0052206 | 0.0441237 | 0.944 |
| | | 0.1 | 0.0046896 | -0.0620751 | 0.948 |
| $\mathcal{M}_{37}$ | 0.005669643 | none | 0.0052634 | 0.0526873 | 0.942 |
| | | 0.1 | 0.0047456 | -0.0508820 | 0.950 |
| $\mathcal{M}_{38}$ | 0.1450273 | none | 0.0036144 | -0.2771174 | 0.938 |
| | | 0.1 | 0.0028068 | -0.4386377 | 0.948 |
| $\mathcal{M}_{39}$ | 0.07251366 | none | 0.0039815 | -0.2037095 | 0.944 |
| | | 0.1 | 0.0031065 | -0.3786992 | 0.938 |

Table 22: Estimation error results: unshared (data1)

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0052460 | 0.0491988 | 0.942 |
| | | 0.1 | 0.0047003 | -0.0599446 | 0.952 |
| $\mathcal{M}_{40}$ | 0.3475735 | none | 0.0053282 | 0.06564517 | 0.948 |
| | | 0.1 | 0.0045461 | -0.0907890 | 0.954 |
| $\mathcal{M}_{41}$ | 0.01737868 | none | 0.0053369 | 0.0673803 | 0.942 |
| | | 0.1 | 0.0045863 | -0.0827344 | 0.954 |
| $\mathcal{M}_{42}$ | 0.01133929 | none | 0.0052251 | 0.0450166 | 0.942 |
| | | 0.1 | 0.0047713 | -0.0457481 | 0.954 |
| $\mathcal{M}_{43}$ | 0.005669643 | none | 0.0052997 | 0.0599378 | 0.944 |
| | | 0.1 | 0.0048422 | -0.0315573 | 0.950 |
| $\mathcal{M}_{44}$ | 0.1450273 | none | 0.0043394 | -0.1321246 | 0.950 |
| | | 0.1 | 0.0041578 | -0.1684345 | 0.946 |
| $\mathcal{M}_{45}$ | 0.07251366 | none | 0.0041840 | -0.1631909 | 0.954 |
| | | 0.1 | 0.0051897 | 0.0379313 | 0.952 |

Table 23: Estimation error results: shared (data1)

# 5 Discussion/Conclusion/Outlook

The results of the simulation studies are summarized in the following.
The approximation error due to rounding does not distort the estimation of the model parameter.
For the unshared assignment error, there are no clear differences between real and observed model coefficients estimators. In the shared and the combined case, there are only very slight differences.
Especially in the cluster version of the generalization error simulation study, strong trends can be seen. The absolute relative bias increases the smaller the variance of the multiplicative error, and the larger the variance of the additive error is.
In the models with assignment and generalization error, differences in estimates between true and observed models partly can be identified, but no clear structure can be noticed.
In the estimation error models, the activity weighting factor turned out to be particularly influential. In the corresponding models, the relative biases differ noticeably from those of the true model.

Since the choice of the true parameters for the generation of the time of death was made based on the exposure values from the data sets, but in some simulations, these exposure values are assumed to be observed values, an inaccuracy is to be expected here. Furthermore, several authors, such as Küchenhoff, Bender, Langner, and Lenz-Tönjes (2003), recommend using a Cox model with a Gompertz distribution to accurately represent the data of the Wismut cohort. This would be another approach that could be pursued.

Another approach that can further be pursued is the estimation error. Instead of true factors per worker, it can be assumed that there are true factors per object or location. The procedure would then be similar to the combination of assignment and generalization error.

# 6 References

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(2), 187–202.

Fahrmeir, P. D. L. (2007). *Lmu statistics, lecture notes: Lebensdauer- und ereignisanalyse.* (Version: SS07)

Hendry, D. J. (2014). Data generation for the cox proportional hazards model with time-dependent covariates: a method for medical researchers. *Statistics in medicine*, *33*(3), 436–454.

Hoffmann, S. (2017). *Approche hiérarchique bayésienne pour la prise en compte d'erreurs de mesure d'exposition chronique et à faible doses aux rayonnements ionisants dans l'estimation du risque de cancers radio-induits: Application à une cohorte de mineurs d'uranium* (Unpublished doctoral dissertation). Paris Saclay.

Kleinbaum, D., & Klein, M. (2012). Survival analysis, springer. *New York*.

Kreuzer, M., Schnelzer, M., Tschense, A., Walsh, L., & Grosche, B. (2009). Cohort profile: the german uranium miners cohort study (wismut cohort), 1946–2003. *International journal of epidemiology*, *39*(4), 980–987.

Küchenhoff, H., Bender, R., Langner, I., & Lenz-Tönjes, R. (2003). *Effect of berkson measurement error on parameter estimates in cox regression models* (Tech. Rep.). Discussion paper//Sonderforschungsbereich 386 der Ludwig-Maximilians ....

Küchenhoff, H., Deffner, V., Aßenmacher, M., Neppl, H., Kaiser, C., Güthlin, D., et al. (2018). Ermittlung der unsicherheiten der strahlenexpositionsabschätzung in der wismut-kohorte-teil i-vorhaben 3616s12223.

# A Appendix

## A.1 More result tables

| model | digits | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | 7 | none | 0.0046151 | -0.0769848 | 0.938 |
| | | 0.1 | 0.0059045 | 0.1809082 | 0.946 |
| $\mathcal{M}_1$ | 0 | none | 0.0046304 | -0.0739104 | 0.942 |
| | | 0.1 | 0.0059153 | 0.1830598 | 0.944 |
| | 1 | none | 0.0046157 | -0.0768692 | 0.938 |
| | | 0.1 | 0.0059048 | 0.1809546 | 0.946 |
| | 2 | none | 0.0046151 | -0.0769782 | 0.938 |
| | | 0.1 | 0.0059044 | 0.1808828 | 0.946 |

Table 24: Approximation error results (data2)

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_2$ | 0.077 | none | 0.0015378 | 0.0008330 | -0.6924340 | -0.8334046 | 0.936 | 0.938 |
| | | 0.1 | 0.0052611 | 0.0064091 | 0.0522198 | 0.2818177 | 0.934 | 0.938 |
| | 0.1 | none | 0.0042778 | 0.0042988 | -0.1444311 | -0.1402317 | 0.940 | 0.954 |
| | | 0.1 | 0.0019382 | 0.0009761 | -0.6123648 | -0.8047794 | 0.968 | 0.968 |
| | 0.8 | none | 0.0041580 | 0.0041177 | -0.1684085 | -0.1764634 | 0.952 | 0.958 |
| | | 0.1 | 0.0032992 | 0.0037915 | -0.3401575 | -0.2416981 | 0.940 | 0.946 |
| $\mathcal{M}_3$ | 190.18 | none | 0.0043092 | 0.0041337 | -0.1381601 | -0.1732664 | 0.952 | 0.958 |
| | | 0.1 | 0.0052000 | 0.0055798 | 0.0399913 | 0.1159644 | 0.944 | 0.946 |
| | 250.15 | none | 0.0031893 | 0.0037990 | -0.3621321 | -0.2401960 | 0.960 | 0.942 |
| | | 0.1 | 0.0072093 | 0.0078573 | 0.4418618 | 0.5714627 | 0.954 | 0.960 |
| | 2721.73 | none | 0.0018037 | 0.0012084 | -0.6392592 | -0.7583300 | 0.952 | 0.952 |
| | | 0.1 | 0.0023639 | 0.0020552 | -0.5272187 | -0.5889635 | 0.948 | 0.944 |
| $\mathcal{M}_4$ | 0.077 | none | 0.0043590 | 0.0038228 | -0.1281953 | -0.2354379 | 0.938 | 0.936 |
| | | 0.1 | 0.0097722 | 0.0101257 | 0.9544365 | 1.0251350 | 0.958 | 0.946 |
| | 0.1 | none | 0.0038289 | 0.0041321 | -0.2342147 | -0.1735849 | 0.932 | 0.932 |
| | | 0.1 | 0.0101371 | 0.0086225 | 1.0274090 | 0.7245015 | 0.972 | 0.966 |
| | 0.8 | none | 0.0038549 | 0.0040910 | -0.2290114 | -0.1817959 | 0.946 | 0.956 |
| | | 0.1 | 0.0049468 | 0.0053885 | -0.0106486 | 0.0776963 | 0.950 | 0.946 |
| $\mathcal{M}_5$ | 190.18 | none | 0.0022369 | 0.0020736 | -0.5526129 | -0.5852893 | 0.954 | 0.956 |
| | | 0.1 | 0.0077743 | 0.0078357 | 0.5548620 | 0.5671300 | 0.956 | 0.972 |
| | 250.15 | none | 0.0059797 | 0.0059506 | 0.1959422 | 0.1901141 | 0.938 | 0.942 |
| | | 0.1 | 0.0075069 | 0.0071342 | 0.5013744 | 0.4268457 | 0.954 | 0.948 |
| | 2721.73 | none | 0.0018726 | 0.0017573 | -0.6254893 | -0.6485405 | 0.950 | 0.942 |
| | | 0.1 | 0.0040480 | 0.0035160 | -0.1904057 | -0.2967909 | 0.946 | 0.960 |

Table 25: Assignment error results: unshared (data2)

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_2$ | 0.077 | none | 0.0049558 | 0.0045560 | -0.0088498 | -0.0888040 | 0.966 | 0.956 |
| | | 0.1 | 0.0030391 | 0.0051033 | -0.3921783 | 0.0206541 | 0.954 | 0.958 |
| | 0.1 | none | 0.0051541 | 0.0046867 | 0.0308275 | -0.0626698 | 0.944 | 0.970 |
| | | 0.1 | 0.0050477 | 0.0056030 | 0.0095493 | 0.1205988 | 0.966 | 0.954 |
| | 0.8 | none | 0.0053335 | 0.0048545 | 0.0667078 | -0.0290972 | 0.946 | 0.950 |
| | | 0.1 | 0.0033057 | 0.0055500 | -0.3388575 | 0.1100090 | 0.960 | 0.958 |
| $\mathcal{M}_3$ | 190.18 | none | 0.0056059 | 0.0051579 | 0.1211805 | 0.0315777 | 0.928 | 0.942 |
| | | 0.1 | 0.0019725 | 0.0026968 | -0.6054950 | -0.4606415 | 0.942 | 0.946 |
| | 250.15 | none | 0.0044137 | 0.0042486 | -0.1172626 | -0.1502720 | 0.972 | 0.956 |
| | | 0.1 | 0.0041746 | 0.0042383 | -0.1650745 | -0.1523362 | 0.952 | 0.964 |
| | 2721.73 | none | 0.0037088 | 0.0030747 | -0.2582339 | -0.3850527 | 0.944 | 0.952 |
| | | 0.1 | 0.0060877 | 0.0083086 | 0.2175477 | 0.6617258 | 0.952 | 0.934 |
| $\mathcal{M}_4$ | 0.077 | none | 0.0033252 | 0.0021895 | -0.3349575 | -0.5621091 | 0.960 | 0.958 |
| | | 0.1 | 0.0065107 | 0.0080597 | 0.3021448 | 0.6119316 | 0.956 | 0.958 |
| | 0.1 | none | 0.0043618 | 0.0052762 | -0.1276391 | 0.0552486 | 0.946 | 0.938 |
| | | 0.1 | 0.0042099 | 0.0076631 | -0.1580130 | 0.5326231 | 0.954 | 0.952 |
| | 0.8 | none | 0.0046189 | 0.0047386 | -0.0762148 | -0.0522739 | 0.950 | 0.952 |
| | | 0.1 | 0.0014902 | 0.0028943 | -0.7019583 | -0.4211373 | 0.960 | 0.958 |
| $\mathcal{M}_5$ | 190.18 | none | 0.0051235 | 0.0050740 | 0.0246984 | 0.0147931 | 0.934 | 0.934 |
| | | 0.1 | 0.0043963 | 0.0048870 | -0.1207390 | -0.0225956 | 0.942 | 0.940 |
| | 250.15 | none | 0.0054290 | 0.0051863 | 0.0857939 | 0.0372617 | 0.950 | 0.956 |
| | | 0.1 | 0.0048252 | 0.0050408 | -0.0349574 | 0.0081509 | 0.960 | 0.966 |
| | 2721.73 | none | 0.0049486 | 0.0043082 | -0.0102864 | -0.1383638 | 0.950 | 0.952 |
| | | 0.1 | 0.0027625 | 0.0028218 | -0.4474915 | -0.4356434 | 0.954 | 0.952 |

Table 26: Assignment error results: shared (data2)

| $\mathcal{M}$ | $\sigma^2$ | censor | $\hat{\beta}_X$ | $\hat{\beta}_Z$ | RB X | RB Z | CR X | CR Z |
|---|---|---|---|---|---|---|---|---|
| $\mathcal{M}_2$ | 0.077 | none | 0.0047040 | 0.0040019 | -0.0592069 | -0.1996208 | 0.964 | 0.962 |
| | | 0.1 | 0.0030199 | 0.0040629 | -0.3960293 | -0.1874120 | 0.958 | 0.952 |
| | 0.1 | none | 0.0047271 | 0.0041885 | -0.0545762 | -0.1622911 | 0.942 | 0.946 |
| | | 0.1 | 0.0022746 | 0.0031574 | -0.5450805 | -0.3685127 | 0.946 | 0.958 |
| | 0.8 | none | 0.0046033 | 0.0045785 | -0.0793487 | -0.0843087 | 0.944 | 0.948 |
| | | 0.1 | -0.0000784 | 0.0013866 | -1.0156870 | -0.7226817 | 0.944 | 0.942 |
| $\mathcal{M}_3$ | 190.18 | none | 0.0053539 | 0.0051130 | 0.0707839 | 0.0225994 | 0.944 | 0.940 |
| | | 0.1 | 0.0036792 | 0.0049064 | -0.2641633 | -0.0187147 | 0.954 | 0.956 |
| | 250.15 | none | 0.0028706 | 0.0013088 | -0.4258878 | -0.7382356 | 0.938 | 0.948 |
| | | 0.1 | 0.0012344 | 0.0007618 | -0.7531144 | -0.8476339 | 0.948 | 0.962 |
| | 2721.73 | none | 0.0056894 | 0.0045818 | 0.1378753 | -0.0836367 | 0.944 | 0.948 |
| | | 0.1 | 0.0051210 | 0.0044177 | 0.0242039 | -0.1164512 | 0.946 | 0.946 |
| $\mathcal{M}_4$ | 0.077 | none | 0.0043106 | 0.0035263 | -0.1378877 | -0.2947378 | 0.922 | 0.926 |
| | | 0.1 | 0.0062306 | 0.0065753 | 0.2461115 | 0.3150654 | 0.954 | 0.954 |
| | 0.1 | none | 0.0041420 | 0.0038247 | -0.1716038 | -0.2350615 | 0.958 | 0.956 |
| | | 0.1 | 0.0023510 | 0.0035971 | -0.5297998 | -0.2805785 | 0.956 | 0.954 |
| | 0.8 | none | 0.0049394 | 0.0044840 | -0.0121130 | -0.1032084 | 0.948 | 0.952 |
| | | 0.1 | 0.0053805 | 0.0072142 | 0.0760992 | 0.4428414 | 0.956 | 0.974 |
| $\mathcal{M}_5$ | 190.18 | none | 0.0018929 | 0.0020937 | -0.6214150 | -0.5812568 | 0.944 | 0.946 |
| | | 0.1 | 0.0064840 | 0.0062147 | 0.2967983 | 0.2429411 | 0.950 | 0.956 |
| | 250.15 | none | 0.0064875 | 0.0060543 | 0.2975007 | 0.2108607 | 0.932 | 0.940 |
| | | 0.1 | 0.0056046 | 0.0046524 | 0.1209165 | -0.0695228 | 0.930 | 0.930 |
| | 2721.73 | none | 0.0027447 | 0.0023312 | -0.4510503 | -0.5337663 | 0.954 | 0.942 |
| | | 0.1 | 0.0033660 | 0.0034542 | -0.3268069 | -0.3091629 | 0.940 | 0.942 |

Table 27: Assignment error results: unshared and shared (data2)

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0051303 | 0.0260612 | 0.958 |
| | | 0.1 | 0.0050321 | 0.0064210 | 0.960 |
| $\mathcal{M}_{14}$ | 0.0008 | none | 0.0055150 | 0.1029948 | 0.948 |
| | | 0.1 | 0.0039464 | -0.2107284 | 0.964 |
| | 0.01 | none | 0.0056552 | 0.1310339 | 0.948 |
| | | 0.1 | 0.0038332 | -0.2333559 | 0.962 |
| | 0.1 | none | 0.0056238 | 0.1247521 | 0.954 |
| | | 0.1 | 0.0040441 | -0.1911736 | 0.964 |
| | 0.8 | none | 0.0054770 | 0.0954059 | 0.952 |
| | | 0.1 | 0.0032351 | -0.3529798 | 0.960 |
| $\mathcal{M}_{15}$ | 5.36 | none | 0.0055233 | 0.1046596 | 0.946 |
| | | 0.1 | 0.0036908 | -0.2618470 | 0.966 |
| | 64.68 | none | 0.0052898 | 0.0579628 | 0.952 |
| | | 0.1 | 0.0041050 | -0.1790023 | 0.966 |
| | 794.31 | none | 0.0047191 | -0.0561843 | 0.958 |
| | | 0.1 | 0.0032107 | -0.3578613 | 0.966 |
| | 5023.42 | none | 0.0031102 | -0.3779648 | 0.952 |
| | | 0.1 | 0.0018738 | -0.6252382 | 0.964 |

Table 28: Generalization error results: cluster (data2)

| model | $\sigma^2$ | censor | $\hat{\beta}$ | RB | CR |
|---|---|---|---|---|---|
| $\mathcal{M}_0$ | none | none | 0.0063497 | 0.2699394 | 0.946 |
| | | 0.1 | 0.0126266 | 1.5253160 | 0.970 |
| $\mathcal{M}_{14}$ | 0.0008 | none | 0.0063453 | 0.2690624 | 0.942 |
| | | 0.1 | 0.0126356 | 1.5271250 | 0.970 |
| | 0.01 | none | 0.0063436 | 0.2687142 | 0.942 |
| | | 0.1 | 0.0123576 | 1.4715290 | 0.970 |
| | 0.1 | none | 0.0064078 | 0.2815538 | 0.950 |
| | | 0.1 | 0.0130145 | 1.6029060 | 0.968 |
| | 0.8 | none | 0.0057424 | 0.1484864 | 0.950 |
| | | 0.1 | 0.0114053 | 1.2810550 | 0.964 |
| $\mathcal{M}_{15}$ | 5.36 | none | 0.0064146 | 0.2829184 | 0.944 |
| | | 0.1 | 0.0126878 | 1.5375570 | 0.970 |
| | 64.68 | none | 0.0064881 | 0.2976245 | 0.942 |
| | | 0.1 | 0.0123827 | 1.4765400 | 0.968 |
| | 794.31 | none | 0.0056409 | 0.1281817 | 0.942 |
| | | 0.1 | 0.0119414 | 1.3882870 | 0.970 |
| | 5023.42 | none | 0.0034269 | -0.3146110 | 0.952 |
| | | 0.1 | 0.0067795 | 0.3559092 | 0.962 |

Table 29: Generalization error results: random (data2)

## A.2 Calculation of error variances

The variances 0.1, 0.8, 5.36, 64.68 and 190.18 were calculated by Küchenhoff et al. (2018). In order to calculate the additive variances corresponding to the multiplicative ones, the following procedure was followed:

For the assignment error, the exposure values of the fictive data set were used as observed exposure. For multiplicative variances, they are then randomly multiplied by a log-normally distributed error with the corresponding variance. Then the variance of the difference between X and Z is calculated. This is used as a variance for the additive errors.
To select the counterpart to the additive errors, the same procedure is performed. It is tried until the desired additive variance is obtained.

For the generalization error, the clustered fictive data is being used. The multiplicative error variance is used to calculate random errors. These are then multiplied by the shaft exposure values to obtain Z. The additive variance then corresponds to the variance of the difference between Z and X.
The calculation of the multiplicative variances is done by trial and error like for the assignment error.

This results in the following variances:

Assignment error:
$0.1 \mathrel{\widehat{=}} 250.15$
$0.8 \mathrel{\widehat{=}} 2721.73$

$190.18 \mathrel{\widehat{=}} 0.077$

Generalization error:
$5.36 \mathrel{\widehat{=}} 0.0008$
$64.68 \mathrel{\widehat{=}} 0.01$

$0.1 \mathrel{\widehat{=}} 794.31$
$0.8 \mathrel{\widehat{=}} 5023.42$

## A.3 Distribution comparison

On the left side, you can always see the distribution in data1 and on the right side the distribution of data2.

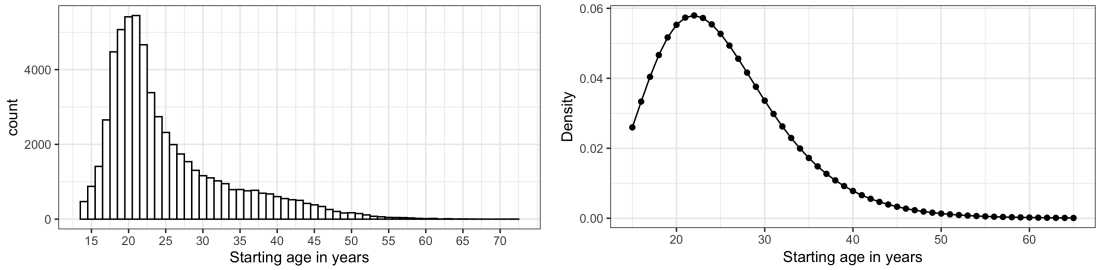


Figure 13: Starting age

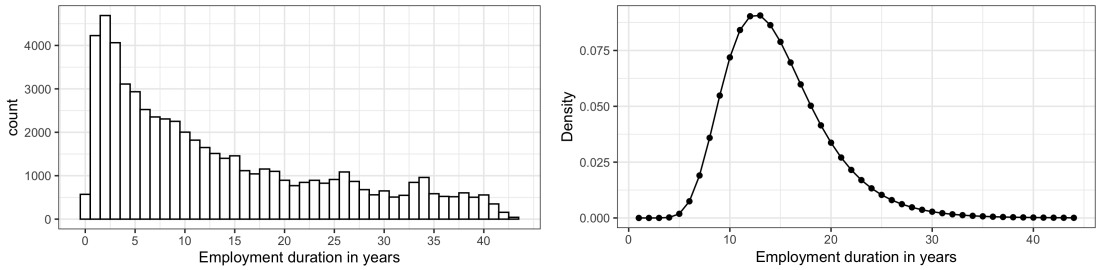


Figure 14: Employment duration

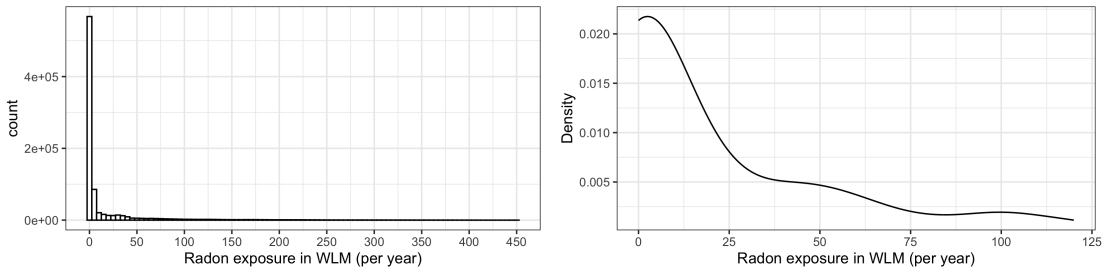


Figure 15: Radon exposure

# A.4 R Code

*Juliana Schäfer*

*September 18, 2019*

## Contents



## General Preparations

### Load packages & set seed

```
library(data.table)
library(dplyr)
library(parallel)
library(foreach)
library(doParallel)
library(msm)
```

```
library(survival)

set.seed(12345)
options(digits = 7)
```

### Preparing parallel computing

```
numCores <- detectCores()
registerDoParallel(numCores)
```

# Data1 (fiktive data of the Wismut cohort)

### Reading in data1

```
data1 <- read.csv("fiktive_daten.txt", sep = "")
setDT(data1)
```

### Adding randomly decimal digits

For the approximation error more than two decimal digits are required. Therefore a noise is applied with jitter() overall exposure values unequal to 0.

```
cols <- paste0("w", 46:89)
data1[, (cols) := lapply(.SD, function(x) ifelse(x != 0, jitter(x), 0)), .SDcols = cols]
```

### Deleting unnecessary columns

The time of death is later regenerated to determine the true coefficients.

```
data1[, c("lung", "yyin_18", "yyout", "wlm") := NULL]
```

### Forming the data set

The dataset is changed so that there are the following columns:

- ID: miner ID
- yyborn: year the miner is born
- yyin: year the miner started working
- yystop: year the miner died
- year: calendar year
- X: exposure in WLM (The values are later sometimes used as true exposure or observed exposure.)
- age: age of the miner

Besides, the lines in which the miner does not yet work or no longer works are deleted.

```
data1 <- melt(data1, id.vars = c("ID", "yyborn", "yyin", "yystop"),
              measure.vars = patterns("^w"),
              variable.name = "year",
              value.name = "X")
```

```r
data1[, year := substr(year, 2, 3)]
data1[, year := as.numeric(paste0("19", year))]

data1 <- data1[year >= yyin & year <= yystop,]

data1[, age := year - yyborn]

setorder(data1, ID)

head(data1, 20)

saveRDS(data1, "data1.RDS")
```

# Data2 (generated data set)

## Distributions of the variables from which is drawn

### Starting age

```
starting_age <- data.table(age = 15:65,
                           p = dlnorm(x = 15:65, meanlog = log(24), sdlog = 0.3))
```

### Employment duration

```
employment_duration <- data.table(duration = 1:44,
                                  p = dlnorm(x = 1:44, meanlog = log(14), sdlog = 0.33))
```

### Radon exposure

```
wlm <- c(20, 40, 35, 40, 50, 55, 65, 80, 100, 120,
         100, 60, 45, 20, 15, 20, 20, 15, 10, 10, 5, 5, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
radon_density <- density(wlm, from = 0, to = 120, n = 1000)
radon_exposure <- data.table(radon = radon_density$x, density = radon_density$y)
radon_exposure[, p_appr := density/sum(radon_exposure$density)]
```

## Data2

Data2 is created with the same amount of miners as data1.

```
set.seed(12345)

data_tmp <- list()
for (i in 1:58974) {
  ID_i <- i
  age_start_i <- sample(x = starting_age$age, size = 1, replace = T, prob = starting_age$p)
  employment_duration_i <- sample(x = employment_duration$duration, size = 1,
                                  replace = T, prob = employment_duration$p)
  yyin_i <- 1946
  yystop_i <- yyin_i + employment_duration_i - 1
  year_i <- yyin_i:yystop_i
  yyborn_i <- yyin_i - age_start_i
  age_i <- age_start_i:(age_start_i + employment_duration_i - 1)
  X_i <- sample(x = radon_exposure$radon, size = length(age_i), replace = T,
                prob = radon_exposure$p_appr)

  data_tmp[[i]] <- data.table(ID = i,
                              yyborn = yyborn_i,
                              yyin = yyin_i,
                              year = year_i,
                              X = X_i,
                              age = age_i)
```

```
}

data2 <- rbindlist(data_tmp)
saveRDS(data2, "data2.RDS")
```

# Approximation error

## Functions

### Generating approximation error

Z_0, Z_1 and Z_2 are the rounded values of the true exposure to 0, 1 and 2 decimal digits. Xcum and the different variants of Zcum are the cumulated X and Z values in WLM. Xcum_100 and Zcum_100 are the cumulated values in WLM/100.

```
approximate_error <- function(data) {
  data[, `:=`(Z_0 = round(X, 0), Z_1 = round(X, 1), Z_2 = round(X, 2))] %>%
    setorder(ID, year)
  data[,`:=`(Xcum = cumsum(X), Zcum_0 = cumsum(Z_0),
             Zcum_1 = cumsum(Z_1), Zcum_2 = cumsum(Z_2)), by = ID]
  data[,`:=`(Xcum_100 = Xcum/100, Zcum_0_100 = Zcum_0/100,
             Zcum_1_100 = Zcum_1/100, Zcum_2_100 = Zcum_2/100)]
}
```

### Generating time of death

Simulate.Survival.Time_appr() creates the time of death for one miner with an approximation error. It is an adapted variant of Hendry's function.

```
Simulate.Survival.Time_appr <-
  function(time.interval, true.X.cum, obs.Z.cum, beta, basehaz, censor = NULL) {

    # CREATING g() AND g^-1()
    g <- function(x) {
      x / basehaz
    }
    g.inv <- function(x) {
      x * basehaz
    }

    #CREATING THE BOUNDS OF TRUNCATION
    t.max <- 100
    t.min <- time.interval[1]

    g.inv.t.max <- g.inv(t.max)
    g.inv.t.min <- g.inv(t.min)

    time.interval <- c(0, time.interval, 110)
    g.inv.t <- g.inv(time.interval)

    X.cum <- c(0, true.X.cum, true.X.cum[length(true.X.cum)])

    obs.Z.cum[[1]] <- c(obs.Z.cum[[1]], obs.Z.cum[[1]][length(obs.Z.cum)])
    obs.Z.cum[[2]] <- c(obs.Z.cum[[2]], obs.Z.cum[[2]][length(obs.Z.cum)])
    obs.Z.cum[[3]] <- c(obs.Z.cum[[3]], obs.Z.cum[[3]][length(obs.Z.cum)])

    lambda <- exp(beta * X.cum)
```

```r
#GENERATING DATA USING ACCEPT-REJECT METHOD
k <- function(x, m, M, rates, t) {
  ifelse(x <= m | x >= M, 0, dpexp(x, rates, t))
}

gen.y <- function(x) {
  r <- 60
  repeat {
    y <- rpexp(r, x, g.inv.t)
    u <- runif(r)
    t <-
      (k(y, g.inv.t.min, g.inv.t.max, x, g.inv.t)) / (dpexp(y, x, g.inv.t))
    y <- y[u <= t][1]
    if (!is.na(y))
      break
  }
  y
}

g.y <-  g(gen.y(lambda))

#CREATING CENSORING INDICATOR
if (is.null(censor)) {
  Y = g.y
} else{
  C <- rexp(1, rate = censor)
  while (C <= t.min) {
    C <- rexp(1, rate = censor)
  }
  Y = min(C, g.y)
}

#CREATING DATASET
data <- data.table(
  start = time.interval[-c(1, which(time.interval >= Y))],
  stop = c(time.interval[-c(1, 2, which(time.interval >= Y))], Y),
  true.cum.exposure =
    X.cum[2:(length(time.interval[-c(1, which(time.interval >= Y))]) + 1)],
  obs.cum.exposure_0 =
    obs.Z.cum[[1]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.cum.exposure_1 =
    obs.Z.cum[[2]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.cum.exposure_2 =
    obs.Z.cum[[3]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  delta = rep(0, length(time.interval[-c(1, which(time.interval >= Y))]))
  )

if (is.null(censor)) {
  data$delta[dim(data)[1]] <- 1
} else if (C > g.y) {
  data$delta[dim(data)[1]] <- 1
}
```

```
    # Against error: Fehler in aeqSurv(Y) : aeqSurv exception, an interval has
    #                effective length 0
    data[stop - start < 0.001, stop := stop + 0.001]

    return(data)
}
```

simulate_delta_appr() randomly selects 500 miners and calculates their time of death. The entire process is repeated 500 times.

```
simulate_delta_appr <-
  function(data_appr, n_rep = 500, n_miners = 500,
           beta = 0.005, basehaz = 0.05, censor = NULL) {

  set.seed(12345)
  seeds <- sample(1:99999, n_rep)

  results <- list()
  for (i in 1:n_rep) {
    set.seed(seeds[i])
    results[[i]] <-
      foreach(j = sample(unique(data_appr$ID), n_miners), .combine = rbind) %do% {

      Simulate.Survival.Time_appr(data_appr[ID == j,]$age,
                                  true.X.cum = data_appr[ID == j,]$Xcum_100,
                                  obs.Z.cum = list(data_appr[ID == j,]$Zcum_0_100,
                                                   data_appr[ID == j,]$Zcum_1_100,
                                                   data_appr[ID == j,]$Zcum_2_100),
                                  beta = beta,
                                  basehaz = basehaz,
                                  censor = censor)
      }

  } %>% setDT()

  return(results)
}
```

simulate_study_appr() fits Cox models and returns various values of interest.

```
simulate_study_appr <- function(data = simulate_delta_appr, beta = 0.005) {

  true_beta <- beta
  coef_X <- c()
  coef_Z_0 <- c()
  coef_Z_1 <- c()
  coef_Z_2 <- c()

  age_mean <- c()
  age_median <- c()

  true.beta.in.CI_X <- rep("No", length(data))
  true.beta.in.CI_Z_0 <- rep("No", length(data))
  true.beta.in.CI_Z_1 <- rep("No", length(data))
  true.beta.in.CI_Z_2 <- rep("No", length(data))
```

```
  for (i in 1:length(data)) {
    setDT(data[[i]])
    model_X <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                     data = data[[i]], control = coxph.control(timefix = FALSE))
    model_Z_0 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0,
                       data = data[[i]], control = coxph.control(timefix = FALSE))
    model_Z_1 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_1,
                       data = data[[i]], control = coxph.control(timefix = FALSE))
    model_Z_2 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_2,
                       data = data[[i]], control = coxph.control(timefix = FALSE))

    coef_X <- c(coef_X, model_X$coefficients)
    coef_Z_0 <- c(coef_Z_0, model_Z_0$coefficients)
    coef_Z_1 <- c(coef_Z_1, model_Z_1$coefficients)
    coef_Z_2 <- c(coef_Z_2, model_Z_2$coefficients)

    age_mean <- c(age_mean, mean(data[[i]][delta == 1,]$stop))
    age_median <- c(age_median, median(data[[i]][delta == 1,]$stop))

    lower_X <- coef(model_X) - qnorm(0.975) * sqrt(model_X$var)
    upper_X <- coef(model_X) + qnorm(0.975) * sqrt(model_X$var)
    if (lower_X < true_beta & true_beta < upper_X) {
      true.beta.in.CI_X[i] <- "yes"
    }

    lower_Z_0 <- coef(model_Z_0) - qnorm(0.975) * sqrt(model_Z_0$var)
    upper_Z_0 <- coef(model_Z_0) + qnorm(0.975) * sqrt(model_Z_0$var)
    if (lower_Z_0 < true_beta & true_beta < upper_Z_0) {
      true.beta.in.CI_Z_0[i] <- "yes"
    }
    lower_Z_1 <- coef(model_Z_1) - qnorm(0.975) * sqrt(model_Z_1$var)
    upper_Z_1 <- coef(model_Z_1) + qnorm(0.975) * sqrt(model_Z_1$var)
    if (lower_Z_1 < true_beta & true_beta < upper_Z_1) {
      true.beta.in.CI_Z_1[i] <- "yes"
    }
    lower_Z_2 <- coef(model_Z_2) - qnorm(0.975) * sqrt(model_Z_2$var)
    upper_Z_2 <- coef(model_Z_2) + qnorm(0.975) * sqrt(model_Z_2$var)
    if (lower_Z_2 < true_beta & true_beta < upper_Z_2) {
      true.beta.in.CI_Z_2[i] <- "yes"
    }

  }

  result <- list(coef_X = coef_X, coef_Z_0 = coef_Z_0,
                 coef_Z_1 = coef_Z_1, coef_Z_2 = coef_Z_2,
                 age_mean = age_mean, age_median = age_median,
                 true.beta.in.CI_X = true.beta.in.CI_X,
                 true.beta.in.CI_Z_0 = true.beta.in.CI_Z_0,
                 true.beta.in.CI_Z_1 = true.beta.in.CI_Z_1,
                 true.beta.in.CI_Z_2 = true.beta.in.CI_Z_2)

  return(result)
}
```

### Data1

```
data1_appr <- approximate_error(data1)

result_data1_appr_delta <- simulate_delta_appr(data_appr = data1_appr, censor = NULL)
saveRDS(result_data1_appr_delta, "appr/result_data1_appr_delta.RDS")

result_data1_appr_sim <- simulate_study_appr(data = result_data1_appr_delta)
saveRDS(result_data1_appr_sim, "appr/result_data1_appr_sim.RDS")

result_data1_appr_delta_censor <- simulate_delta_appr(data_appr = data1_appr, censor = 0.1)
saveRDS(result_data1_appr_delta_censor, "appr/result_data1_appr_delta_censor.RDS")

result_data1_appr_sim_censor <- simulate_study_appr(data = result_data1_appr_delta_censor)
saveRDS(result_data1_appr_sim_censor, "appr/result_data1_appr_sim_censor.RDS")
```

### Data2

```
data2_appr <- approximate_error(data2)

result_data2_appr_delta <- simulate_delta_appr(data_appr = data2_appr, censor = NULL)
saveRDS(result_data2_appr_delta, "appr/result_data2_appr_delta.RDS")

result_data2_appr_sim <- simulate_study_appr(data = result_data2_appr_delta)
saveRDS(result_data2_appr_sim, "appr/result_data2_appr_sim.RDS")

result_data2_appr_delta_censor <- simulate_delta_appr(data_appr = data2_appr, censor = 0.1)
saveRDS(result_data2_appr_delta_censor, "appr/result_data2_appr_delta_censor.RDS")

result_data2_appr_sim_censor <- simulate_study_appr(data = result_data2_appr_delta_censor)
saveRDS(result_data2_appr_sim_censor, "appr/result_data2_appr_sim_censor.RDS")
```

# Assignment error

## Functions

This function creates errors that is log-normally or normally distributed. The variance is passed through var, n is the amount of errors to be generated, and multiplicative decides whether a log-normally or a normally distribution is used.

```
assignment_u <- function(var, n, multiplicative) {

  if (multiplicative == T) {
    U <- rlnorm(meanlog = -var/2, sdlog = sqrt(var), n = n)
  }

  else if (multiplicative == F) {
    U <- rnorm(mean = 0, sd = sqrt(var), n = n)
  }

  return(U)
}
```

### Generating assignment error

assignment_error() creates X values for different kinds of assignment errors. If X $< 0$, X is set to 0.

```
assignment_error <-
  function(shared = "unshared", homoscedastic = T, multiplicative = T, var, data) {

  ##############################################################################
  ########################### unshared, homoscedastic ##########################
  # Error per worker and year, all equal variance

  if (shared == "unshared" & homoscedastic == T) {
    # Generating U
    data[, U := assignment_u(var, n = nrow(data),
                             multiplicative = multiplicative)]
  }

  ##############################################################################
  ########################### unshared, heteroscedastic ########################
  # Error per worker and year, different variance per shaft

  else if (shared == "unshared" & homoscedastic == F) {
    # Generating var_j for each shaft
    vars <- data.table(shaft = 1:length(unique(data$shaft)),
                       var_j = rnorm(mean = var,
                                     sd = if_else(multiplicative == T, 0.05, 10),
                                     n = length(unique(data$shaft)))
                       )

    # var_j shouldn't be negative
    while (nrow(vars[var_j < 0,]) != 0) {
      vars[var_j < 0, var_j := rnorm(mean = var,
                                     sd = if_else(multiplicative == T, 0.05, 10),
```

```
                                    n = length(nrow(vars[var_j < 0,])))]
  }

  # Combining miners with corresponding var_j
  data <- na.omit(data[vars, on = "shaft", nomatch = 0]) %>% setorder(ID, year)

  # Generating U
  data[, U := assignment_u(var_j, n = nrow(data), multiplicative = multiplicative)]
  }

################################################################################
############################ shared, homoscedastic #############################
# Error per worker, all equal variance

else if (shared == "shared" & homoscedastic == T) {
  # Generating U
  data[, U := rep(assignment_u(var, n = 1, multiplicative = multiplicative)), by = ID]
}

################################################################################
########################### shared, heteroscedastic ############################
# Error per worker, different variance per shaft

else if (shared == "shared" & homoscedastic == F) {

  # Generating var_j for each shaft
  vars <- data.table(shaft = 1:length(unique(data$shaft)),
                     var_j = rnorm(mean = var,
                                   sd = if_else(multiplicative == T, 0.05, 10),
                                   n = length(unique(data$shaft)))
  )

  # var_j shouldn't be negative
  while (nrow(vars[var_j < 0,]) != 0) {
    vars[var_j < 0, var_j := rnorm(mean = var,
                                   sd = if_else(multiplicative == T, 0.05, 10),
                                   n = length(nrow(vars[var_j < 0,])))]
  }

   # Combining miners with corresponding var_j
  data <- na.omit(data[vars, on = "shaft", nomatch = 0]) %>% setorder(ID, year)

  # Generate U
  data[, U := assignment_u(var_j, n = 1, multiplicative = multiplicative), by = ID]
} # end else if

################################################################################
####################### unshared, shared, homoscedastic ########################

else if (shared == "both" & homoscedastic == T) {

  # Generating U unshared: Uu
  data[, Uu := assignment_u(var, n = nrow(data),
```

```
                              multiplicative = multiplicative)]

  # Generating U shared: Us
  data[, Us := rep(assignment_u(var, n = 1, multiplicative = multiplicative)),
       by = ID]

}

 ##############################################################################
 ###################### unshared, shared, heteroscedastic ######################

 else if (shared == "both" & homoscedastic == F) {

   # Generating var_j for each shaft
   vars <- data.table(shaft = 1:length(unique(data$shaft)),
                      var_j = rnorm(mean = var,
                                    sd = if_else(multiplicative == T, 0.05, 10),
                                    n = length(unique(data$shaft)))
   )

   # var_j shouldn't be negative
   while (nrow(vars[var_j < 0,]) != 0) {
     vars[var_j < 0, var_j := rnorm(mean = var,
                                    sd = if_else(multiplicative == T, 0.05, 10),
                                    n = length(nrow(vars[var_j < 0,])))]
   }

   # Combining miners with corresponding var_j
   data <- na.omit(data[vars, on = "shaft", nomatch = 0]) %>% setorder(ID, year)

   # Generating Uu
   data[, Uu := assignment_u(var_j, n = nrow(data), multiplicative = multiplicative)]

   # Generating Us
   data[, Us := assignment_u(var_j, n = 1, multiplicative = multiplicative), by = ID]

 } # end else if

##############################################################################
################################## in general ################################

if (shared != "both") {

  # Calculate X
  if (multiplicative == T) {
    data[, X := Z * U]
  }

  else if (multiplicative == F) {
    data[, X := Z + U]
  }

} # end if (shared)
```

```
  else {
    # Calculating X
    if (multiplicative == T) {
      data[, X := Z * Uu * Us]
    }

    else if (multiplicative == F) {
      data[, X := Z + Uu + Us]
    }
  } # end else

  data[X < 0, X := 0]

  # Calculating Xcum
  data[, Xcum := cumsum(X), by = ID]

  # Calculating Xcum_100
  data[, Xcum_100 := Xcum/100]

  setorder(data, ID, year)

  return(data$Xcum_100)
} # end function
```

### Generating time of death

Simulate.Survival.Time_assign() creates the time of death for one miner with an assignment error. It is an adapted variant of Hendry's function.

```
Simulate.Survival.Time_assign <-
  function(time.interval, true.X.cum, obs.Z.cum, beta, basehaz, censor = NULL, type) {

    # CREATING g() AND g^-1()
    g <- function(x) {
      x / basehaz
    }
    g.inv <- function(x) {
      x * basehaz
    }

    #CREATING THE BOUNDS OF TRUNCATION
    t.max <- 100
    t.min <- time.interval[1]

    g.inv.t.max <- g.inv(t.max)
    g.inv.t.min <- g.inv(t.min)

    time.interval <- c(0, time.interval, 110)
    g.inv.t <- g.inv(time.interval)

    obs.Z.cum <- c(obs.Z.cum, obs.Z.cum[length(obs.Z.cum)])

    data <- list()
```

```r
X.cum <- list()
for (i in 1:length(true.X.cum)) {
  X.cum[[i]] <- c(0, true.X.cum[[i]], true.X.cum[[i]][length(true.X.cum[[i]])])
}

lambda <- list()
for (i in 1:length(true.X.cum)) {
  lambda[[i]] <- exp(beta * X.cum[[i]])
}


#GENERATING DATA USING ACCEPT-REJECT METHOD
k <- function(x, m, M, rates, t) {
  ifelse(x <= m | x >= M, 0, dpexp(x, rates, t))
}

gen.y <- function(x) {
  r <- 60
  repeat {
    y <- rpexp(r, x, g.inv.t)
    u <- runif(r)
    t <-
      (k(y, g.inv.t.min, g.inv.t.max, x, g.inv.t)) / (dpexp(y, x, g.inv.t))
    y <- y[u <= t][1]
    if (!is.na(y))
      break
  }
  y
}

g.y <- list()
for (i in 1:length(true.X.cum)) {
  g.y[[i]] <-  g(gen.y(lambda[[i]]))
}


#CREATING CENSORING INDICATOR
if (is.null(censor)) {
  Y <- list()
  for (i in 1:length(true.X.cum)) {
    Y[[i]] <-  g.y[[i]]
  }
} else {
  Y <- list()
  C <- rexp(1, rate = censor)
  while (C <= t.min) {
    C <- rexp(1, rate = censor)
  }
  for (i in 1:length(true.X.cum)) {
    Y[[i]] <-  min(C, g.y[[i]])
  }
}
```

```
    #CREATING DATASET
    data <- list()
    for (i in 1:length(true.X.cum)) {
      data[[i]] <- data.table(
        start = time.interval[-c(1, which(time.interval >= Y[[i]]))],
        stop = c(time.interval[-c(1, 2, which(time.interval >= Y[[i]]))], Y[[i]]),
        true.cum.exposure = X.cum[[i]][2:(length(
          time.interval[-c(1, which(time.interval >= Y[[i]]))]) + 1)],
        obs.cum.exposure = obs.Z.cum[1:length(
          time.interval[-c(1, which(time.interval >= Y[[i]]))])],
        delta = rep(0, length(time.interval[-c(1, which(time.interval >= Y[[i]]))])))


    if (is.null(censor)) {
      data[[i]]$delta[dim(data[[i]])[1]] <- 1
    } else if (C > g.y[[i]]) {
      data[[i]]$delta[dim(data[[i]])[1]] <- 1
    }

    # Against error: Fehler in aeqSurv(Y) :
    # aeqSurv exception, an interval has effective length 0

    data[[i]][stop - start < 0.001, stop := stop + 0.001]

    }

    if (type == "ho_mu") {
      names(data) <- c("Xcum_100_ho_mu_0.077",
                       "Xcum_100_ho_mu_0.1",
                       "Xcum_100_ho_mu_0.8")
    }
    if (type == "ho_ad") {
      names(data) <- c("Xcum_100_ho_ad_190.18",
                       "Xcum_100_ho_ad_250.15",
                       "Xcum_100_ho_ad_2721.73")
    }
    if (type == "he_mu") {
      names(data) <- c("Xcum_100_he_mu_0.077",
                       "Xcum_100_he_mu_0.1",
                       "Xcum_100_he_mu_0.8")
    }
    if (type == "he_ad") {
      names(data) <- c("Xcum_100_he_ad_190.18",
                       "Xcum_100_he_ad_250.15",
                       "Xcum_100_he_ad_2721.73")
    }

    return(data)
}
```

simulate_delta_assign() randomly selects 500 miners and calculates their time of death. The entire process is repeated 500 times.

```r
simulate_delta_assign <-
  function(data_assign, n_rep = 500, n_miners = 500, shared = T, type,
           beta = 0.005, basehaz = 0.05, censor = NULL) {

  set.seed(12345)
  seeds <- sample(1:99999, n_rep)

  results <- list()
  for (i in 1:n_rep) {
    set.seed(seeds[i])

    if (type == "ho_ad") {
      Xcum_list <- list()
    }
    if (type == "he_mu") {
      Xcum_list <- list()
    }
    if (type == "he_ad") {
      Xcum_list <- list()
    }

    if (type == "ho_mu") {
    results[[i]] <- foreach(j = sample(unique(data_assign$ID), n_miners)) %do% {
      Simulate.Survival.Time_assign(data_assign[ID == j,]$age,
                                    true.X.cum = list(
                                      data_assign[ID == j,]$Xcum_100_ho_mu_0.077,
                                      data_assign[ID == j,]$Xcum_100_ho_mu_0.1,
                                      data_assign[ID == j,]$Xcum_100_ho_mu_0.8),
                                    obs.Z.cum = data_assign[ID == j,]$Zcum_100,
                                    beta = beta,
                                    basehaz = basehaz,
                                    censor = censor,
                                    type = "ho_mu")
    }
    }

    if (type == "ho_ad") {
    results[[i]] <- foreach(j = sample(unique(data_assign$ID), n_miners)) %do% {
      Simulate.Survival.Time_assign(data_assign[ID == j,]$age,
                                    true.X.cum = list(
                                      data_assign[ID == j,]$Xcum_100_ho_ad_190.18,
                                      data_assign[ID == j,]$Xcum_100_ho_ad_250.15,
                                      data_assign[ID == j,]$Xcum_100_ho_ad_2721.73),
                                    obs.Z.cum = data_assign[ID == j,]$Zcum_100,
                                    beta = beta,
                                    basehaz = basehaz,
                                    censor = censor,
                                    type = "ho_ad")
    }
    }

    if (type == "he_mu") {
    results[[i]] <- foreach(j = sample(unique(data_assign$ID), n_miners)) %do% {
```

```
      Simulate.Survival.Time_assign(data_assign[ID == j,]$age,
                                    true.X.cum = list(
                                      data_assign[ID == j,]$Xcum_100_he_mu_0.077,
                                      data_assign[ID == j,]$Xcum_100_he_mu_0.1,
                                      data_assign[ID == j,]$Xcum_100_he_mu_0.8),
                                    obs.Z.cum = data_assign[ID == j,]$Zcum_100,
                                    beta = beta,
                                    basehaz = basehaz,
                                    censor = censor,
                                    type = "he_mu")
      }
    }

    if (type == "he_ad") {
    results[[i]] <- foreach(j = sample(unique(data_assign$ID), n_miners)) %do% {
      Simulate.Survival.Time_assign(data_assign[ID == j,]$age,
                                    true.X.cum = list(
                                      data_assign[ID == j,]$Xcum_100_he_ad_190.18,
                                      data_assign[ID == j,]$Xcum_100_he_ad_250.15,
                                      data_assign[ID == j,]$Xcum_100_he_ad_2721.73),
                                    obs.Z.cum = data_assign[ID == j,]$Zcum_100,
                                    beta = beta,
                                    basehaz = basehaz,
                                    censor = censor,
                                    type = "he_ad")
      }
    }

  }

  names(results) <- paste0("rep", 1:n_rep)

  results_final <- list()

  for (k in names(results)) {
    results_final[[paste0(k)]] <- list()
    length(results_final[[paste0(k)]]) <- 3
    names(results_final[[paste0(k)]]) <- names(results[[1]][[1]])
  }
  for (k in 1:n_rep) {
    for (l in 1:3) {
      results_final[[k]][[l]] <- rbindlist(sapply(results[[k]], function(x) x[l]))
    }
  }

  return(results_final)
}
```

simulate_study_assign() fits Cox models and returns various values of interest.

```
simulate_study_assign <- function(data = simulate_delta_assign, beta = 0.005) {

  true_beta <- beta
  coef_X_var1 <- c()
```

```r
coef_X_var2 <- c()
coef_X_var3 <- c()
coef_Z_var1 <- c()
coef_Z_var2 <- c()
coef_Z_var3 <- c()

age_mean_var1 <- c()
age_mean_var2 <- c()
age_mean_var3 <- c()
age_median_var1 <- c()
age_median_var2 <- c()
age_median_var3 <- c()

true.beta.in.CI_X_var1 <- rep("No", length(data))
true.beta.in.CI_X_var2 <- rep("No", length(data))
true.beta.in.CI_X_var3 <- rep("No", length(data))
true.beta.in.CI_Z_var1 <- rep("No", length(data))
true.beta.in.CI_Z_var2 <- rep("No", length(data))
true.beta.in.CI_Z_var3 <- rep("No", length(data))

for (i in 1:length(data)) {
  model_X_var1 <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                        data = data[[i]][[1]], control = coxph.control(timefix = F))
  model_X_var2 <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                        data = data[[i]][[2]], control = coxph.control(timefix = F))
  model_X_var3 <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                        data = data[[i]][[3]], control = coxph.control(timefix = F))

  model_Z_var1 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure,
                        data = data[[i]][[1]], control = coxph.control(timefix = F))
  model_Z_var2 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure,
                        data = data[[i]][[2]], control = coxph.control(timefix = F))
  model_Z_var3 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure,
                        data = data[[i]][[3]], control = coxph.control(timefix = F))

  coef_X_var1 <- c(coef_X_var1, model_X_var1$coefficients)
  coef_X_var2 <- c(coef_X_var2, model_X_var2$coefficients)
  coef_X_var3 <- c(coef_X_var3, model_X_var3$coefficients)

  coef_Z_var1 <- c(coef_Z_var1, model_Z_var1$coefficients)
  coef_Z_var2 <- c(coef_Z_var2, model_Z_var2$coefficients)
  coef_Z_var3 <- c(coef_Z_var3, model_Z_var3$coefficients)

  age_mean_var1 <- c(age_mean_var1, mean(data[[i]][[1]][delta == 1,]$stop))
  age_mean_var2 <- c(age_mean_var2, mean(data[[i]][[2]][delta == 1,]$stop))
  age_mean_var3 <- c(age_mean_var3, mean(data[[i]][[3]][delta == 1,]$stop))
  age_median_var1 <- c(age_median_var1, median(data[[i]][[1]][delta == 1,]$stop))
  age_median_var2 <- c(age_median_var2, median(data[[i]][[2]][delta == 1,]$stop))
  age_median_var3 <- c(age_median_var3, median(data[[i]][[3]][delta == 1,]$stop))

  lower_X_var1 <- coef(model_X_var1) - qnorm(0.975) * sqrt(model_X_var1$var)
  upper_X_var1 <- coef(model_X_var1) + qnorm(0.975) * sqrt(model_X_var1$var)
  if (lower_X_var1 < true_beta & true_beta < upper_X_var1) {
```

```
    true.beta.in.CI_X_var1[i] <- "yes"
  }
  lower_X_var2 <- coef(model_X_var2) - qnorm(0.975) * sqrt(model_X_var2$var)
  upper_X_var2 <- coef(model_X_var2) + qnorm(0.975) * sqrt(model_X_var2$var)
  if (lower_X_var2 < true_beta & true_beta < upper_X_var2) {
    true.beta.in.CI_X_var2[i] <- "yes"
  }
  lower_X_var3 <- coef(model_X_var3) - qnorm(0.975) * sqrt(model_X_var3$var)
  upper_X_var3 <- coef(model_X_var3) + qnorm(0.975) * sqrt(model_X_var3$var)
  if (lower_X_var3 < true_beta & true_beta < upper_X_var3) {
    true.beta.in.CI_X_var3[i] <- "yes"
  }

  lower_Z_var1 <- coef(model_Z_var1) - qnorm(0.975) * sqrt(model_Z_var1$var)
  upper_Z_var1 <- coef(model_Z_var1) + qnorm(0.975) * sqrt(model_Z_var1$var)
  if (lower_Z_var1 < true_beta & true_beta < upper_Z_var1) {
    true.beta.in.CI_Z_var1[i] <- "yes"
  }
  lower_Z_var2 <- coef(model_Z_var2) - qnorm(0.975) * sqrt(model_Z_var2$var)
  upper_Z_var2 <- coef(model_Z_var2) + qnorm(0.975) * sqrt(model_Z_var2$var)
  if (lower_Z_var2 < true_beta & true_beta < upper_Z_var2) {
    true.beta.in.CI_Z_var2[i] <- "yes"
  }
  lower_Z_var3 <- coef(model_Z_var3) - qnorm(0.975) * sqrt(model_Z_var3$var)
  upper_Z_var3 <- coef(model_Z_var3) + qnorm(0.975) * sqrt(model_Z_var3$var)
  if (lower_Z_var3 < true_beta & true_beta < upper_Z_var3) {
    true.beta.in.CI_Z_var3[i] <- "yes"
  }

}

result <- list(coef_X_var1 = coef_X_var1,
               coef_X_var2 = coef_X_var2,
               coef_X_var3 = coef_X_var3,
               coef_Z_var1 = coef_Z_var1,
               coef_Z_var2 = coef_Z_var2,
               coef_Z_var3 = coef_Z_var3,

               age_mean_var1 = age_mean_var1,
               age_mean_var2 = age_mean_var2,
               age_mean_var3 = age_mean_var3,

               age_median_var1 = age_median_var1,
               age_median_var2 = age_median_var2,
               age_median_var3 = age_median_var3,

               true.beta.in.CI_X_var1 = true.beta.in.CI_X_var1,
               true.beta.in.CI_X_var2 = true.beta.in.CI_X_var2,
               true.beta.in.CI_X_var3 = true.beta.in.CI_X_var3,

               true.beta.in.CI_Z_var1 = true.beta.in.CI_Z_var1,
               true.beta.in.CI_Z_var2 = true.beta.in.CI_Z_var2,
               true.beta.in.CI_Z_var3 = true.beta.in.CI_Z_var3
```

```
                )

  return(result)
}
```

## Data

save_results_assign() calculates the corresponding assignment error, fits Cox models, and returns values of interest.

```
save_results_assign <- function(data, data_string) {

  set.seed(12345)

  data_assign <- data

  setnames(data_assign, c("X"), c("Z"))

  data_assign[, Zcum := cumsum(Z), by = ID]
  data_assign[, Zcum_100 := Zcum/100]
  data_assign[, shaft := sample(1:10, length(unique(ID)), replace = T), by = ID]

  for (i in c("unshared", "shared", "both")) {
    assign(paste0("data_assign_", i),
           cbind(data_assign,
                 Xcum_100_ho_mu_0.077 = assignment_error(shared = i, homoscedastic = T,
                                                         multiplicative = T,
                                                         var = 0.077, data),
                 Xcum_100_ho_mu_0.1 = assignment_error(shared = i, homoscedastic = T,
                                                       multiplicative = T,
                                                       var = 0.077, data),
                 Xcum_100_ho_mu_0.8 = assignment_error(shared = i, homoscedastic = T,
                                                       multiplicative = T,
                                                       var = 0.077, data),
                 Xcum_100_ho_ad_190.18 = assignment_error(shared = i, homoscedastic = T,
                                                          multiplicative = T,
                                                          var = 0.077, data),
                 Xcum_100_ho_ad_250.15 = assignment_error(shared = i, homoscedastic = T,
                                                          multiplicative = T,
                                                          var = 0.077, data),
                 Xcum_100_ho_ad_2721.73 = assignment_error(shared = i, homoscedastic = T,
                                                           multiplicative = T,
                                                           var = 0.077, data),
                 Xcum_100_he_mu_0.077 = assignment_error(shared = i, homoscedastic = T,
                                                         multiplicative = T,
                                                         var = 0.077, data),
                 Xcum_100_he_mu_0.1 = assignment_error(shared = i, homoscedastic = T,
                                                       multiplicative = T,
                                                       var = 0.077, data),
                 Xcum_100_he_mu_0.8 = assignment_error(shared = i, homoscedastic = T,
                                                       multiplicative = T,
                                                       var = 0.077, data),
                 Xcum_100_he_ad_190.18 = assignment_error(shared = i, homoscedastic = T,
```

```
                                                    multiplicative = T,
                                                    var = 0.077, data),
                Xcum_100_he_ad_250.15 = assignment_error(shared = i, homoscedastic = T,
                                                    multiplicative = T,
                                                    var = 0.077, data),
                Xcum_100_he_ad_2721.73 = assignment_error(shared = i, homoscedastic = T,
                                                     multiplicative = T,
                                                     var = 0.077, data)
                )
    )
  }


  for (i in c("both")) {
    for (j in c("ho_ad", "he_mu", "he_ad")) {

      # without censor
      assign(paste0("result_", data_string, "_assign_delta_", i, "_", j),
             simulate_delta_assign(get(paste0("data_assign_", i)), type = j))
      saveRDS(get(paste0("result_", data_string, "_assign_delta_", i, "_", j)),
              paste0("assign/result_", data_string, "_assign_delta_", i, "_", j, ".RDS"))

      # with censor
      assign(paste0("result_", data_string, "_assign_delta_", i, "_", j, "_censor"),
             simulate_delta_assign(get(paste0("data_assign_", i)),
                                   type = j, censor = 0.1))
      saveRDS(get(paste0("result_", data_string, "_assign_delta_", i, "_", j, "_censor")),
              paste0("assign/result_", data_string, "_assign_delta_", i, "_", j, "_censor.RDS"))
    }
  }

  for (i in c("unshared", "shared", "both")) {
    for (j in c("ho_mu", "ho_ad", "he_mu", "he_ad")) {

      # without censor
      assign(paste0("result_", data_string, "_assign_sim_", i, "_", j),
             simulate_study_assign(get(paste0(
               "result_", data_string, "_assign_delta_", i, "_", j))))
      saveRDS(get(paste0("result_", data_string, "_assign_sim_", i, "_", j)),
              paste0("assign/result_", data_string, "_assign_sim_", i, "_", j, ".RDS"))

      # with censor
      assign(paste0("result_", data_string, "_assign_sim_", i, "_", j, "_censor"),
             simulate_study_assign(get(paste0(
               "result_", data_string, "_assign_delta_", i, "_", j, "_censor"))))
      saveRDS(get(paste0("result_", data_string, "_assign_sim_", i, "_", j, "_censor")),
              paste0("assign/result_", data_string, "_assign_sim_", i, "_", j, "_censor.RDS"))
    }
  }
}
```

```
save_results_assign(data1, "data1")
save_results_assign(data2, "data2")
```

# Generalization error

## Functions

cluster() randomly divides the workers into 5 mines and within each mine into 5 clusters.

```
cluster <- function(data) {
  data[, Xmean := mean(X), by = "ID"]
  data_cluster <- unique(data, by = c("ID", "Xmean"))
  data_cluster[, mine := sample(1:5, nrow(data_cluster), replace = T)]
  fit <- list()
  result <- list()
  for (i in 1:5) {
    fit[[i]] <- kmeans(data_cluster[mine == i]$Xmean, centers = 5)
    aggregate(data_cluster[mine == i], by = list(fit[[i]]$cluster), FUN = mean)
    result[[i]] <- data.table(data_cluster[mine == i], shaft = fit[[i]]$cluster)
  }
  data_cluster <- rbindlist(result)
  result <- data[data_cluster[,.(ID, mine, shaft)], on = .(ID = ID)]
}
```

### Generating time of death

Simulate.Survival.Time_gener() creates the time of death for one miner with an generalization error. It is an adapted variant of Hendry's function.

```
Simulate.Survival.Time_gener <-
  function(time.interval, true.X.cum, obs.Z.cum, beta, basehaz, censor = NULL) {

    # CREATING g() AND g^-1()
    g <- function(x) {
      x / basehaz
    }
    g.inv <- function(x) {
      x * basehaz
    }

    #CREATING THE BOUNDS OF TRUNCATION
    t.max <- 100
    t.min <- time.interval[1]

    g.inv.t.max <- g.inv(t.max)
    g.inv.t.min <- g.inv(t.min)

    time.interval <- c(0, time.interval, 110)
    g.inv.t <- g.inv(time.interval)

    X.cum <- c(0, true.X.cum, true.X.cum[length(true.X.cum)])

    obs.Z.cum[[1]] <- c(obs.Z.cum[[1]], obs.Z.cum[[1]][length(obs.Z.cum)])
    obs.Z.cum[[2]] <- c(obs.Z.cum[[2]], obs.Z.cum[[2]][length(obs.Z.cum)])
    obs.Z.cum[[3]] <- c(obs.Z.cum[[3]], obs.Z.cum[[3]][length(obs.Z.cum)])
    obs.Z.cum[[4]] <- c(obs.Z.cum[[4]], obs.Z.cum[[4]][length(obs.Z.cum)])
    obs.Z.cum[[5]] <- c(obs.Z.cum[[5]], obs.Z.cum[[5]][length(obs.Z.cum)])
```

```r
obs.Z.cum[[6]] <- c(obs.Z.cum[[6]], obs.Z.cum[[6]][length(obs.Z.cum)])
obs.Z.cum[[7]] <- c(obs.Z.cum[[7]], obs.Z.cum[[7]][length(obs.Z.cum)])
obs.Z.cum[[8]] <- c(obs.Z.cum[[8]], obs.Z.cum[[8]][length(obs.Z.cum)])

lambda <- exp(beta * X.cum)

#GENERATING DATA USING ACCEPT-REJECT METHOD
k <- function(x, m, M, rates, t) {
  ifelse(x <= m | x >= M, 0, dpexp(x, rates, t))
}

gen.y <- function(x) {
  r <- 60
  repeat {
    y <- rpexp(r, x, g.inv.t)
    u <- runif(r)
    t <-
      (k(y, g.inv.t.min, g.inv.t.max, x, g.inv.t)) / (dpexp(y, x, g.inv.t))
    y <- y[u <= t][1]
    if (!is.na(y))
      break
  }
  y
}

g.y <-  g(gen.y(lambda))

#CREATING CENSORING INDICATOR
if (is.null(censor)) {
  Y = g.y
} else{
  C <- rexp(1, rate = censor)
  while (C <= t.min) {
    C <- rexp(1, rate = censor)
  }
  Y = min(C, g.y)
}
#CREATING DATASET
data <- data.table(
  time.interval[-c(1, which(time.interval >= Y))],
  c(time.interval[-c(1, 2, which(time.interval >= Y))], Y),
  X.cum[2:(length(time.interval[-c(1, which(time.interval >= Y))]) + 1)],
  obs.Z.cum[[1]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[2]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[3]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[4]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[5]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[6]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[7]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[8]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  rep(0, length(time.interval[-c(1, which(time.interval >= Y))]))
  )
```

```
    colnames(data) <-
      c("start",
        "stop",
        "true.cum.exposure",
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[1]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[2]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[3]],"[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[4]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[5]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[6]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[7]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_",
               stringr::str_extract(names(obs.Z.cum)[[8]], "[:digit:]{1,}\\.[:digit:]{1,}")),
        "delta")

    if (is.null(censor)) {
      data$delta[dim(data)[1]] <- 1
    } else if (C > g.y) {
      data$delta[dim(data)[1]] <- 1
    }


    # Against error: Fehler in aeqSurv(Y) :
    # aeqSurv exception, an interval has effective length 0
    data[stop - start < 0.001, stop := stop + 0.001]

    return(data)

}
```

simulate_delta_gener() randomly selects 500 miners and calculates their time of death. The entire process is repeated 500 times.

```
simulate_delta_gener <- function(data = data, n_rep = 500, n_miners = 500,
                                 beta = 0.005, basehaz = 0.05, censor = NULL) {

  set.seed(12345)
  seeds <- sample(1:99999, n_rep)

  results <- list()
  for (i in 1:n_rep) {
    set.seed(seeds[i])
    results[[i]] <-
      foreach(j = sample(unique(data$ID), n_miners), .combine = rbind) %do% {

      Simulate.Survival.Time_gener(
        time.interval = data[ID == j,]$age,
```

```
        true.X.cum = data[ID == j,]$Xcum_100,
        obs.Z.cum = list(
          Zcum_100_object_mu_0.0008 = data[ID == j,]$Zcum_100_object_mu_0.0008,
          Zcum_100_object_mu_0.01 = data[ID == j,]$Zcum_100_object_mu_0.01,
          Zcum_100_object_mu_0.1 = data[ID == j,]$Zcum_100_object_mu_0.1,
          Zcum_100_object_mu_0.8 = data[ID == j,]$Zcum_100_object_mu_0.8,
          Zcum_100_object_ad_5.36 = data[ID == j,]$Zcum_100_object_ad_5.36,
          Zcum_100_object_ad_64.68 = data[ID == j,]$Zcum_100_object_ad_64.68,
          Zcum_100_object_ad_794.31 = data[ID == j,]$Zcum_100_object_ad_794.31,
          Zcum_100_object_ad_5023.42 = data[ID == j,]$Zcum_100_object_ad_5023.42),
        beta = beta,
        basehaz = basehaz,
        censor = censor)
      }

  }
  return(results)
}
```

simulate_study_gener() fits Cox models and returns various values of interest.

```
simulate_study_gener <- function(data = simulate_delta_gener, beta = 0.005) {

  true_beta <- beta
  coef_X <- c()
  coef_Z_0.0008 <- c()
  coef_Z_0.01 <- c()
  coef_Z_0.1 <- c()
  coef_Z_0.8 <- c()
  coef_Z_5.36 <- c()
  coef_Z_64.68 <- c()
  coef_Z_794.31 <- c()
  coef_Z_5023.42 <- c()

  age_mean <- c()
  age_median <- c()

  true.beta.in.CI_X <- rep("No", length(data))
  true.beta.in.CI_Z_0.0008 <- rep("No", length(data))
  true.beta.in.CI_Z_0.01 <- rep("No", length(data))
  true.beta.in.CI_Z_0.1 <- rep("No", length(data))
  true.beta.in.CI_Z_0.8 <- rep("No", length(data))
  true.beta.in.CI_Z_5.36 <- rep("No", length(data))
  true.beta.in.CI_Z_64.68 <- rep("No", length(data))
  true.beta.in.CI_Z_794.31 <- rep("No", length(data))
  true.beta.in.CI_Z_5023.42 <- rep("No", length(data))

  for (i in 1:length(data)) {
    setDT(data[[i]])
    model_X <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                     data = data[[i]], control = coxph.control(timefix = FALSE))
    model_Z_0.0008 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.0008,
                            data = data[[i]], control = coxph.control(timefix = FALSE))
    model_Z_0.01 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.01,
```

```
                        data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_0.1 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.1,
                     data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_0.8 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.8,
                     data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_5.36 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_5.36,
                      data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_64.68 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_64.68,
                       data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_794.31 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_794.31,
                        data = data[[i]], control = coxph.control(timefix = FALSE))
model_Z_5023.42 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_5023.42,
                         data = data[[i]], control = coxph.control(timefix = FALSE))

coef_X <- c(coef_X, model_X$coefficients)
coef_Z_0.0008 <- c(coef_Z_0.0008, model_Z_0.0008$coefficients)
coef_Z_0.01 <- c(coef_Z_0.01, model_Z_0.01$coefficients)
coef_Z_0.1 <- c(coef_Z_0.1, model_Z_0.1$coefficients)
coef_Z_0.8 <- c(coef_Z_0.8, model_Z_0.8$coefficients)
coef_Z_5.36 <- c(coef_Z_5.36, model_Z_5.36$coefficients)
coef_Z_64.68 <- c(coef_Z_64.68, model_Z_64.68$coefficients)
coef_Z_794.31 <- c(coef_Z_794.31, model_Z_794.31$coefficients)
coef_Z_5023.42 <- c(coef_Z_5023.42, model_Z_5023.42$coefficients)

age_mean <- c(age_mean, mean(data[[i]][delta == 1,]$stop))
age_median <- c(age_median, median(data[[i]][delta == 1,]$stop))

lower_X <- coef(model_X) - qnorm(0.975) * sqrt(model_X$var)
upper_X <- coef(model_X) + qnorm(0.975) * sqrt(model_X$var)
if (lower_X < true_beta & true_beta < upper_X) {
  true.beta.in.CI_X[i] <- "yes"
}

lower_Z_0.0008 <- coef(model_Z_0.0008) - qnorm(0.975) * sqrt(model_Z_0.0008$var)
upper_Z_0.0008 <- coef(model_Z_0.0008) + qnorm(0.975) * sqrt(model_Z_0.0008$var)
if (lower_Z_0.0008 < true_beta & true_beta < upper_Z_0.0008) {
  true.beta.in.CI_Z_0.0008[i] <- "yes"
}
lower_Z_0.01 <- coef(model_Z_0.01) - qnorm(0.975) * sqrt(model_Z_0.01$var)
upper_Z_0.01 <- coef(model_Z_0.01) + qnorm(0.975) * sqrt(model_Z_0.01$var)
if (lower_Z_0.01 < true_beta & true_beta < upper_Z_0.01) {
  true.beta.in.CI_Z_0.01[i] <- "yes"
}
lower_Z_0.1 <- coef(model_Z_0.1) - qnorm(0.975) * sqrt(model_Z_0.1$var)
upper_Z_0.1 <- coef(model_Z_0.1) + qnorm(0.975) * sqrt(model_Z_0.1$var)
if (lower_Z_0.1 < true_beta & true_beta < upper_Z_0.1) {
  true.beta.in.CI_Z_0.1[i] <- "yes"
}
lower_Z_0.8 <- coef(model_Z_0.8) - qnorm(0.975) * sqrt(model_Z_0.8$var)
upper_Z_0.8 <- coef(model_Z_0.8) + qnorm(0.975) * sqrt(model_Z_0.8$var)
if (lower_Z_0.8 < true_beta & true_beta < upper_Z_0.8) {
  true.beta.in.CI_Z_0.8[i] <- "yes"
}
```

```
    lower_Z_5.36 <- coef(model_Z_5.36) - qnorm(0.975) * sqrt(model_Z_5.36$var)
    upper_Z_5.36 <- coef(model_Z_5.36) + qnorm(0.975) * sqrt(model_Z_5.36$var)
    if (lower_Z_5.36 < true_beta & true_beta < upper_Z_5.36) {
      true.beta.in.CI_Z_5.36[i] <- "yes"
    }
    lower_Z_64.68 <- coef(model_Z_64.68) - qnorm(0.975) * sqrt(model_Z_64.68$var)
    upper_Z_64.68 <- coef(model_Z_64.68) + qnorm(0.975) * sqrt(model_Z_64.68$var)
    if (lower_Z_64.68 < true_beta & true_beta < upper_Z_64.68) {
      true.beta.in.CI_Z_64.68[i] <- "yes"
    }
    lower_Z_794.31 <- coef(model_Z_794.31) - qnorm(0.975) * sqrt(model_Z_794.31$var)
    upper_Z_794.31 <- coef(model_Z_794.31) + qnorm(0.975) * sqrt(model_Z_794.31$var)
    if (lower_Z_794.31 < true_beta & true_beta < upper_Z_794.31) {
      true.beta.in.CI_Z_794.31[i] <- "yes"
    }
    lower_Z_5023.42 <- coef(model_Z_5023.42) - qnorm(0.975) * sqrt(model_Z_5023.42$var)
    upper_Z_5023.42 <- coef(model_Z_5023.42) + qnorm(0.975) * sqrt(model_Z_5023.42$var)
    if (lower_Z_5023.42 < true_beta & true_beta < upper_Z_5023.42) {
      true.beta.in.CI_Z_5023.42[i] <- "yes"
    }

  }

  result <- list(coef_X = coef_X,
                 coef_Z_0.0008 = coef_Z_0.0008,
                 coef_Z_0.01 = coef_Z_0.01,
                 coef_Z_0.1 = coef_Z_0.1,
                 coef_Z_0.8 = coef_Z_0.8,
                 coef_Z_5.36 = coef_Z_5.36,
                 coef_Z_64.68 = coef_Z_64.68,
                 coef_Z_794.31 = coef_Z_794.31,
                 coef_Z_5023.42 = coef_Z_5023.42,
                 age_mean = age_mean,
                 age_median = age_median,
                 true.beta.in.CI_X = true.beta.in.CI_X,
                 true.beta.in.CI_Z_0.0008 = true.beta.in.CI_Z_0.0008,
                 true.beta.in.CI_Z_0.01 = true.beta.in.CI_Z_0.01,
                 true.beta.in.CI_Z_0.1 = true.beta.in.CI_Z_0.1,
                 true.beta.in.CI_Z_0.8 = true.beta.in.CI_Z_0.8,
                 true.beta.in.CI_Z_5.36 = true.beta.in.CI_Z_5.36,
                 true.beta.in.CI_Z_64.68 = true.beta.in.CI_Z_64.68,
                 true.beta.in.CI_Z_794.31 = true.beta.in.CI_Z_794.31,
                 true.beta.in.CI_Z_5023.42 = true.beta.in.CI_Z_5023.42
                 )

  return(result)
}
```

## Data1

### Cluster

If Z is negative, Z is set to zero.

```
set.seed(12345)
data1_cluster <- cluster(data1)
data1_cluster[, Xshaft := mean(Xmean), by = c("mine", "shaft", "year")]

data1_cluster[, `:=`(U_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                     U_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                     U_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                     U_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                     U_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                     U_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                     U_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                     U_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                     ),
              by = c("mine", "shaft", "year")]

data1_cluster[, `:=`(Z_mu_0.0008 = ifelse(Xshaft * U_mu_0.0008 > 0, Xshaft * U_mu_0.0008, 0),
                     Z_mu_0.01 = ifelse(Xshaft * U_mu_0.01 > 0, Xshaft * U_mu_0.01, 0),
                     Z_mu_0.1 = ifelse(Xshaft * U_mu_0.1 > 0, Xshaft * U_mu_0.1, 0),
                     Z_mu_0.8 = ifelse(Xshaft * U_mu_0.8 > 0, Xshaft * U_mu_0.8, 0),
                     Z_ad_5.36 = ifelse(Xshaft + U_ad_5.36 > 0, Xshaft + U_ad_5.36, 0),
                     Z_ad_64.68 = ifelse(Xshaft + U_ad_64.68 > 0, Xshaft + U_ad_64.68, 0),
                     Z_ad_794.31 = ifelse(Xshaft + U_ad_794.31 > 0, Xshaft + U_ad_794.31, 0),
                     Z_ad_5023.42 = ifelse(Xshaft + U_ad_5023.42 > 0, Xshaft + U_ad_5023.42, 0)
                     )]

data1_cluster[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                     Zobject_mu_0.01 = mean(Z_mu_0.01),
                     Zobject_mu_0.1 = mean(Z_mu_0.1),
                     Zobject_mu_0.8 = mean(Z_mu_0.8),
                     Zobject_ad_5.36 = mean(Z_ad_5.36),
                     Zobject_ad_64.68 = mean(Z_ad_64.68),
                     Zobject_ad_794.31 = mean(Z_ad_794.31),
                     Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]

data1_cluster[, `:=`(Xcum_100 = cumsum(Xshaft)/100,
                     Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                     Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                     Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                     Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                     Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                     Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                     Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                     Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                     ),
              by = ID]

result_data1_gener_delta_cluster <- simulate_delta_gener(data = data1_cluster)
```

```
saveRDS(result_data1_gener_delta_cluster, "gener/result_data1_gener_delta_cluster.RDS")

result_data1_gener_delta_cluster_censor <-
  simulate_delta_gener(data = data1_cluster, censor = 0.1)
saveRDS(result_data1_gener_delta_cluster_censor,
        "gener/result_data1_gener_delta_cluster_censor.RDS")

result_data1_gener_sim_cluster <-
  simulate_study_gener(result_data1_gener_delta_cluster)
saveRDS(result_data1_gener_sim_cluster, "gener/result_data1_gener_sim_cluster.RDS")

result_data1_gener_sim_cluster_censor <- simulate_study_gener(result_data1_gener_delta_cluster_censor)
saveRDS(result_data1_gener_sim_cluster_censor,
        "gener/result_data1_gener_sim_cluster_censor.RDS")
```

### Random

```
set.seed(12345)
data1_random <- data1[, `:=`(mine = sample(1:5, 1, replace = T),
                             shaft = sample(1:5, 1, replace = T)),
                      by = ID]

data1_random[, Xshaft := mean(X), by = c("mine", "shaft", "year")]

data1_random[, `:=`(U_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                    U_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                    U_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                    U_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                    U_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                    U_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                    U_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                    U_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                    ),
             by = c("mine", "shaft", "year")]

data1_random[, `:=`(Z_mu_0.0008 = ifelse(Xshaft * U_mu_0.0008 > 0, Xshaft * U_mu_0.0008, 0),
                    Z_mu_0.01 = ifelse(Xshaft * U_mu_0.01 > 0, Xshaft * U_mu_0.01, 0),
                    Z_mu_0.1 = ifelse(Xshaft * U_mu_0.1 > 0, Xshaft * U_mu_0.1, 0),
                    Z_mu_0.8 = ifelse(Xshaft * U_mu_0.8 > 0, Xshaft * U_mu_0.8, 0),
                    Z_ad_5.36 = ifelse(Xshaft + U_ad_5.36 > 0, Xshaft + U_ad_5.36, 0),
                    Z_ad_64.68 = ifelse(Xshaft + U_ad_64.68 > 0, Xshaft + U_ad_64.68, 0),
                    Z_ad_794.31 = ifelse(Xshaft + U_ad_794.31 > 0, Xshaft + U_ad_794.31, 0),
                    Z_ad_5023.42 = ifelse(Xshaft + U_ad_5023.42 > 0, Xshaft + U_ad_5023.42, 0)
                    )]

data1_random[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                    Zobject_mu_0.01 = mean(Z_mu_0.01),
                    Zobject_mu_0.1 = mean(Z_mu_0.1),
                    Zobject_mu_0.8 = mean(Z_mu_0.8),
                    Zobject_ad_5.36 = mean(Z_ad_5.36),
                    Zobject_ad_64.68 = mean(Z_ad_64.68),
                    Zobject_ad_794.31 = mean(Z_ad_794.31),
```

```
                    Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]

data1_random[, `:=`(Xcum_100 = cumsum(Xshaft)/100,
                    Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                    Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                    Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                    Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                    Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                    Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                    Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                    Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                    ),
              by = ID]

result_data1_gener_delta_random <- simulate_delta_gener(data = data1_random)
saveRDS(result_data1_gener_delta_random, "gener/result_data1_gener_delta_random.RDS")

result_data1_gener_delta_random_censor <-
  simulate_delta_gener(data = data1_random, censor = 0.1)
saveRDS(result_data1_gener_delta_random_censor,
        "gener/result_data1_gener_delta_random_censor.RDS")

result_data1_gener_sim_random <- simulate_study_gener(result_data1_gener_delta_random)
saveRDS(result_data1_gener_sim_random, "gener/result_data1_gener_sim_random.RDS")

result_data1_gener_sim_random_censor <-
  simulate_study_gener(result_data1_gener_delta_random_censor)
saveRDS(result_data1_gener_sim_random_censor,
        "gener/result_data1_gener_sim_random_censor.RDS")
```

## Data2

### Cluster

If Z is negative, Z is set to zero.

```
set.seed(12345)
data2_cluster <- cluster(data2)
data2_cluster[, Xshaft := mean(Xmean), by = c("mine", "shaft", "year")]

data2_cluster[, `:=`(U_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                    U_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                    U_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                    U_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                    U_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                    U_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                    U_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                    U_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                    ),
              by = c("mine", "shaft", "year")]

data2_cluster[, `:=`(Z_mu_0.0008 = ifelse(Xshaft * U_mu_0.0008 > 0, Xshaft * U_mu_0.0008, 0),
```

```
                      Z_mu_0.01 = ifelse(Xshaft * U_mu_0.01 > 0, Xshaft * U_mu_0.01, 0),
                      Z_mu_0.1 = ifelse(Xshaft * U_mu_0.1 > 0, Xshaft * U_mu_0.1, 0),
                      Z_mu_0.8 = ifelse(Xshaft * U_mu_0.8 > 0, Xshaft * U_mu_0.8, 0),
                      Z_ad_5.36 = ifelse(Xshaft + U_ad_5.36 > 0, Xshaft + U_ad_5.36, 0),
                      Z_ad_64.68 = ifelse(Xshaft + U_ad_64.68 > 0, Xshaft + U_ad_64.68, 0),
                      Z_ad_794.31 = ifelse(Xshaft + U_ad_794.31 > 0, Xshaft + U_ad_794.31, 0),
                      Z_ad_5023.42 = ifelse(Xshaft + U_ad_5023.42 > 0, Xshaft + U_ad_5023.42, 0)
                      )]

data2_cluster[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                      Zobject_mu_0.01 = mean(Z_mu_0.01),
                      Zobject_mu_0.1 = mean(Z_mu_0.1),
                      Zobject_mu_0.8 = mean(Z_mu_0.8),
                      Zobject_ad_5.36 = mean(Z_ad_5.36),
                      Zobject_ad_64.68 = mean(Z_ad_64.68),
                      Zobject_ad_794.31 = mean(Z_ad_794.31),
                      Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]

data2_cluster[, `:=`(Xcum_100 = cumsum(Xshaft)/100,
                      Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                      Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                      Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                      Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                      Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                      Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                      Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                      Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                      ),
              by = ID]

result_data2_gener_delta_cluster <- simulate_delta_gener(data = data2_cluster)
saveRDS(result_data2_gener_delta_cluster, "gener/result_data2_gener_delta_cluster.RDS")

result_data2_gener_delta_cluster_censor <-
  simulate_delta_gener(data = data2_cluster, censor = 0.1)
saveRDS(result_data2_gener_delta_cluster_censor,
        "gener/result_data2_gener_delta_cluster_censor.RDS")

result_data2_gener_sim_cluster <- simulate_study_gener(result_data2_gener_delta_cluster)
saveRDS(result_data2_gener_sim_cluster, "gener/result_data2_gener_sim_cluster.RDS")

result_data2_gener_sim_cluster_censor <-
  simulate_study_gener(result_data2_gener_delta_cluster_censor)
saveRDS(result_data2_gener_sim_cluster_censor,
        "gener/result_data2_gener_sim_cluster_censor.RDS")
```

### Random

```
set.seed(12345)
data2_random <- data2[, `:=`(mine = sample(1:5, 1, replace = T),
                             shaft = sample(1:5, 1, replace = T)),
```

```
                      by = ID]

data2_random[, Xshaft := mean(X), by = c("mine", "shaft", "year")]

data2_random[, `:=`(U_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                    U_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                    U_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                    U_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                    U_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                    U_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                    U_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                    U_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                    ),
               by = c("mine", "shaft", "year")]

data2_random[, `:=`(Z_mu_0.0008 = ifelse(Xshaft * U_mu_0.0008 > 0, Xshaft * U_mu_0.0008, 0),
                    Z_mu_0.01 = ifelse(Xshaft * U_mu_0.01 > 0, Xshaft * U_mu_0.01, 0),
                    Z_mu_0.1 = ifelse(Xshaft * U_mu_0.1 > 0, Xshaft * U_mu_0.1, 0),
                    Z_mu_0.8 = ifelse(Xshaft * U_mu_0.8 > 0, Xshaft * U_mu_0.8, 0),
                    Z_ad_5.36 = ifelse(Xshaft + U_ad_5.36 > 0, Xshaft + U_ad_5.36, 0),
                    Z_ad_64.68 = ifelse(Xshaft + U_ad_64.68 > 0, Xshaft + U_ad_64.68, 0),
                    Z_ad_794.31 = ifelse(Xshaft + U_ad_794.31 > 0, Xshaft + U_ad_794.31, 0),
                    Z_ad_5023.42 = ifelse(Xshaft + U_ad_5023.42 > 0, Xshaft + U_ad_5023.42, 0)
                    )]

data2_random[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                    Zobject_mu_0.01 = mean(Z_mu_0.01),
                    Zobject_mu_0.1 = mean(Z_mu_0.1),
                    Zobject_mu_0.8 = mean(Z_mu_0.8),
                    Zobject_ad_5.36 = mean(Z_ad_5.36),
                    Zobject_ad_64.68 = mean(Z_ad_64.68),
                    Zobject_ad_794.31 = mean(Z_ad_794.31),
                    Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
               by = c("year", "mine")]

data2_random[, `:=`(Xcum_100 = cumsum(Xshaft)/100,
                    Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                    Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                    Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                    Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                    Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                    Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                    Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                    Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                    ),
               by = ID]

result_data2_gener_delta_random <- simulate_delta_gener(data = data2_random)
saveRDS(result_data2_gener_delta_random, "gener/result_data2_gener_delta_random.RDS")

result_data2_gener_delta_random_censor <-
  simulate_delta_gener(data = data2_random, censor = 0.1)
saveRDS(result_data2_gener_delta_random_censor,
```

```
        "gener/result_data2_gener_delta_random_censor.RDS")

result_data2_gener_sim_random <- simulate_study_gener(result_data2_gener_delta_random)
saveRDS(result_data2_gener_sim_random, "gener/result_data2_gener_sim_random.RDS")

result_data2_gener_sim_random_censor <-
  simulate_study_gener(result_data2_gener_delta_random_censor)
saveRDS(result_data2_gener_sim_random_censor,
        "gener/result_data2_gener_sim_random_censor.RDS")
```

# Assignment and generalization error

## Functions

cluster() randomly divides the workers into 5 mines and within each mine into 5 clusters.

```
cluster <- function(data) {
  data[, Xmean := mean(X), by = "ID"]
  data_cluster <- unique(data, by = c("ID", "Xmean"))
  data_cluster[, mine := sample(1:5, nrow(data_cluster), replace = T)]
  fit <- list()
  result <- list()
  for (i in 1:5) {
    fit[[i]] <- kmeans(data_cluster[mine == i]$Xmean, centers = 5)
    aggregate(data_cluster[mine == i], by = list(fit[[i]]$cluster), FUN = mean)
    result[[i]] <- data.table(data_cluster[mine == i], shaft = fit[[i]]$cluster)
  }
  data_cluster <- rbindlist(result)
  result <- data[data_cluster[,.(ID, mine, shaft)], on = .(ID = ID)]
}
```

### Generating time of death

Simulate.Survival.Time_assign_gener() creates the time of death for one miner with an assignment and generalization error. It is an adapted variant of Hendry's function.

```
Simulate.Survival.Time_assign_gener <-
  function(time.interval, true.X.cum, obs.Z.cum, beta, basehaz, censor = NULL) {

    # CREATING g() AND g^-1()
    g <- function(x) {
      x / basehaz
    }
    g.inv <- function(x) {
      x * basehaz
    }

    #CREATING THE BOUNDS OF TRUNCATION
    t.max <- 100
    t.min <- time.interval[1]

    g.inv.t.max <- g.inv(t.max)
    g.inv.t.min <- g.inv(t.min)

    time.interval <- c(0, time.interval, 110)
    g.inv.t <- g.inv(time.interval)

    X.cum <- c(0, true.X.cum, true.X.cum[length(true.X.cum)])

    obs.Z.cum[[1]] <- c(obs.Z.cum[[1]], obs.Z.cum[[1]][length(obs.Z.cum)])
    obs.Z.cum[[2]] <- c(obs.Z.cum[[2]], obs.Z.cum[[2]][length(obs.Z.cum)])
    obs.Z.cum[[3]] <- c(obs.Z.cum[[3]], obs.Z.cum[[3]][length(obs.Z.cum)])
    obs.Z.cum[[4]] <- c(obs.Z.cum[[4]], obs.Z.cum[[4]][length(obs.Z.cum)])
```

```
lambda <- exp(beta * X.cum)

#GENERATING DATA USING ACCEPT-REJECT METHOD
k <- function(x, m, M, rates, t) {
  ifelse(x <= m | x >= M, 0, dpexp(x, rates, t))
}

gen.y <- function(x) {
  r <- 60
  repeat {
    y <- rpexp(r, x, g.inv.t)
    u <- runif(r)
    t <-
      (k(y, g.inv.t.min, g.inv.t.max, x, g.inv.t)) / (dpexp(y, x, g.inv.t))
    y <- y[u <= t][1]
    if (!is.na(y))
      break
  }
  y
}

g.y <-  g(gen.y(lambda))

#CREATING CENSORING INDICATOR
if (is.null(censor)) {
  Y = g.y
} else{
  C <- rexp(1, rate = censor)
  while (C <= t.min) {
    C <- rexp(1, rate = censor)
  }
  Y = min(C, g.y)
}
#CREATING DATASET
data <- data.table(
  time.interval[-c(1, which(time.interval >= Y))],
  c(time.interval[-c(1, 2, which(time.interval >= Y))], Y),
  X.cum[2:(length(time.interval[-c(1, which(time.interval >= Y))]) + 1)],
  obs.Z.cum[[1]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[2]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[3]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[4]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  rep(0, length(time.interval[-c(1, which(time.interval >= Y))]))
  )

colnames(data) <-
  c("start",
    "stop",
    "true.cum.exposure",
    paste0("obs.cum.exposure_", stringr::str_extract(names(obs.Z.cum)[[1]],
           "[:digit:]{1,}\\.[:digit:]{1,}")),
    paste0("obs.cum.exposure_", stringr::str_extract(names(obs.Z.cum)[[2]],
           "[:digit:]{1,}\\.[:digit:]{1,}")),
```

```
        paste0("obs.cum.exposure_", stringr::str_extract(names(obs.Z.cum)[[3]],
               "[:digit:]{1,}\\.[:digit:]{1,}")),
        paste0("obs.cum.exposure_", stringr::str_extract(names(obs.Z.cum)[[4]],
               "[:digit:]{1,}\\.[:digit:]{1,}")),
        "delta")

    if (is.null(censor)) {
      data$delta[dim(data)[1]] <- 1
    } else if (C > g.y) {
      data$delta[dim(data)[1]] <- 1
    }

    #setorder(data, ID, year)

    # Against error: Fehler in aeqSurv(Y) :
    # aeqSurv exception, an interval has effective length 0
    data[stop - start < 0.001, stop := stop + 0.001]

    return(data)

}
```

simulate_delta_assign_gener() randomly selects 500 miners and calculates their time of death. The entire process is repeated 500 times.

```
simulate_delta_assign_gener <-
  function(data = data, n_rep = 500, n_miners = 500,
           multiplicative, Xcum_100, beta = 0.005, basehaz = 0.05, censor = NULL) {

  set.seed(12345)
  seeds <- sample(1:99999, n_rep)

  results <- list()
  for (i in 1:n_rep) {
    set.seed(seeds[i])
    results[[i]] <- foreach(j = sample(unique(data$ID), n_miners), .combine = rbind) %do% {

      if (multiplicative == T)  {

      Simulate.Survival.Time_assign_gener(
        data[ID == j,]$age,
        true.X.cum = data[ID == j,][[Xcum_100]],
        obs.Z.cum = list(
          Zcum_100_object_mu_0.0008 = data[ID == j,]$Zcum_100_object_mu_0.0008,
          Zcum_100_object_mu_0.01 = data[ID == j,]$Zcum_100_object_mu_0.01,
          Zcum_100_object_mu_0.1 = data[ID == j,]$Zcum_100_object_mu_0.1,
          Zcum_100_object_mu_0.8 = data[ID == j,]$Zcum_100_object_mu_0.8
          ),
        beta = beta,
        basehaz = basehaz,
        censor = censor) %>%
          setDT()
        }
```

```
    else if (multiplicative == F) {

      Simulate.Survival.Time_assign_gener(
        data[ID == j,]$age,
        true.X.cum = data[ID == j,][[Xcum_100]],
        obs.Z.cum = list(
          Zcum_100_object_ad_5.36 = data[ID == j,]$Zcum_100_object_ad_5.36,
          Zcum_100_object_ad_64.68 = data[ID == j,]$Zcum_100_object_ad_64.68,
          Zcum_100_object_ad_794.31 = data[ID == j,]$Zcum_100_object_ad_794.31,
          Zcum_100_object_ad_5023.42 = data[ID == j,]$Zcum_100_object_ad_5023.42
          ),
        beta = beta,
        basehaz = basehaz,
        censor = censor) %>%
        setDT()
      }

    }
}
  return(results)
}
```

simulate_study_assign_gener() fits Cox models and returns various values of interest.

```
simulate_study_assign_gener <-
  function(data = simulate_delta, multiplicative, beta = 0.005) {

  true_beta <- beta
  coef_X <- c()

  age_mean <- c()
  age_median <- c()

  true.beta.in.CI_X <- rep("No", length(data))

  if (multiplicative == T) {
    coef_Z_0.0008 <- c()
    coef_Z_0.01 <- c()
    coef_Z_0.1 <- c()
    coef_Z_0.8 <- c()
    true.beta.in.CI_Z_0.0008 <- rep("No", length(data))
    true.beta.in.CI_Z_0.01 <- rep("No", length(data))
    true.beta.in.CI_Z_0.1 <- rep("No", length(data))
    true.beta.in.CI_Z_0.8 <- rep("No", length(data))
  }
  else if (multiplicative == F) {
    coef_Z_5.36 <- c()
    coef_Z_64.68 <- c()
    coef_Z_794.31 <- c()
    coef_Z_5023.42 <- c()
    true.beta.in.CI_Z_5.36 <- rep("No", length(data))
    true.beta.in.CI_Z_64.68 <- rep("No", length(data))
    true.beta.in.CI_Z_794.31 <- rep("No", length(data))
    true.beta.in.CI_Z_5023.42 <- rep("No", length(data))
```

```
}

for (i in 1:length(data)) {
  setDT(data[[i]])
  model_X <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                   data = data[[i]], control = coxph.control(timefix = F))
  coef_X <- c(coef_X, model_X$coefficients)
  age_mean <- c(age_mean, mean(data[[i]][delta == 1,]$stop))
  age_median <- c(age_median, median(data[[i]][delta == 1,]$stop))

  lower_X <- coef(model_X) - qnorm(0.975) * sqrt(model_X$var)
  upper_X <- coef(model_X) + qnorm(0.975) * sqrt(model_X$var)
  if (lower_X < true_beta & true_beta < upper_X) {
    true.beta.in.CI_X[i] <- "yes"
  }

  if (multiplicative == T) {
    model_Z_0.0008 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.0008,
                            data = data[[i]], control = coxph.control(timefix = F))
    model_Z_0.01 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.01,
                          data = data[[i]], control = coxph.control(timefix = F))
    model_Z_0.1 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.1,
                         data = data[[i]], control = coxph.control(timefix = F))
    model_Z_0.8 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_0.8,
                         data = data[[i]], control = coxph.control(timefix = F))
    coef_Z_0.0008 <- c(coef_Z_0.0008, model_Z_0.0008$coefficients)
    coef_Z_0.01 <- c(coef_Z_0.01, model_Z_0.01$coefficients)
    coef_Z_0.1 <- c(coef_Z_0.1, model_Z_0.1$coefficients)
    coef_Z_0.8 <- c(coef_Z_0.8, model_Z_0.8$coefficients)

    lower_Z_0.0008 <- coef(model_Z_0.0008) - qnorm(0.975) * sqrt(model_Z_0.0008$var)
    upper_Z_0.0008 <- coef(model_Z_0.0008) + qnorm(0.975) * sqrt(model_Z_0.0008$var)
    if (lower_Z_0.0008 < true_beta & true_beta < upper_Z_0.0008) {
      true.beta.in.CI_Z_0.0008[i] <- "yes"
    }

    lower_Z_0.01 <- coef(model_Z_0.01) - qnorm(0.975) * sqrt(model_Z_0.01$var)
    upper_Z_0.01 <- coef(model_Z_0.01) + qnorm(0.975) * sqrt(model_Z_0.01$var)
    if (lower_Z_0.01 < true_beta & true_beta < upper_Z_0.01) {
      true.beta.in.CI_Z_0.01[i] <- "yes"
    }

    lower_Z_0.1 <- coef(model_Z_0.1) - qnorm(0.975) * sqrt(model_Z_0.1$var)
    upper_Z_0.1 <- coef(model_Z_0.1) + qnorm(0.975) * sqrt(model_Z_0.1$var)
    if (lower_Z_0.1 < true_beta & true_beta < upper_Z_0.1) {
      true.beta.in.CI_Z_0.1[i] <- "yes"
    }

    lower_Z_0.8 <- coef(model_Z_0.8) - qnorm(0.975) * sqrt(model_Z_0.8$var)
    upper_Z_0.8 <- coef(model_Z_0.8) + qnorm(0.975) * sqrt(model_Z_0.8$var)
    if (lower_Z_0.8 < true_beta & true_beta < upper_Z_0.8) {
      true.beta.in.CI_Z_0.8[i] <- "yes"
    }
```

```
  }

  else if (multiplicative == F) {
    model_Z_5.36 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_5.36,
                          data = data[[i]], control = coxph.control(timefix = F))
    model_Z_64.68 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_64.68,
                           data = data[[i]], control = coxph.control(timefix = F))
    model_Z_794.31 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_794.31,
                            data = data[[i]], control = coxph.control(timefix = F))
    model_Z_5023.42 <- coxph(Surv(start, stop, delta) ~ obs.cum.exposure_5023.42,
                             data = data[[i]], control = coxph.control(timefix = F))
    coef_Z_5.36 <- c(coef_Z_5.36, model_Z_5.36$coefficients)
    coef_Z_64.68 <- c(coef_Z_64.68, model_Z_64.68$coefficients)
    coef_Z_794.31 <- c(coef_Z_794.31, model_Z_794.31$coefficients)
    coef_Z_5023.42 <- c(coef_Z_5023.42, model_Z_5023.42$coefficients)

    lower_Z_5.36 <- coef(model_Z_5.36) - qnorm(0.975) * sqrt(model_Z_5.36$var)
    upper_Z_5.36 <- coef(model_Z_5.36) + qnorm(0.975) * sqrt(model_Z_5.36$var)
    if (lower_Z_5.36 < true_beta & true_beta < upper_Z_5.36) {
      true.beta.in.CI_Z_5.36[i] <- "yes"
    }

    lower_Z_64.68 <- coef(model_Z_64.68) - qnorm(0.975) * sqrt(model_Z_64.68$var)
    upper_Z_64.68 <- coef(model_Z_64.68) + qnorm(0.975) * sqrt(model_Z_64.68$var)
    if (lower_Z_64.68 < true_beta & true_beta < upper_Z_64.68) {
      true.beta.in.CI_Z_64.68[i] <- "yes"
    }

    lower_Z_794.31 <- coef(model_Z_794.31) - qnorm(0.975) * sqrt(model_Z_794.31$var)
    upper_Z_794.31 <- coef(model_Z_794.31) + qnorm(0.975) * sqrt(model_Z_794.31$var)
    if (lower_Z_794.31 < true_beta & true_beta < upper_Z_794.31) {
      true.beta.in.CI_Z_794.31[i] <- "yes"
    }

    lower_Z_5023.42 <- coef(model_Z_5023.42) - qnorm(0.975) * sqrt(model_Z_5023.42$var)
    upper_Z_5023.42 <- coef(model_Z_5023.42) + qnorm(0.975) * sqrt(model_Z_5023.42$var)
    if (lower_Z_5023.42 < true_beta & true_beta < upper_Z_5023.42) {
      true.beta.in.CI_Z_5023.42[i] <- "yes"
    }

  }
}

result <- if (multiplicative == T) {
  list(coef_X = coef_X,
       coef_Z_0.0008 = coef_Z_0.0008,
       coef_Z_0.01 = coef_Z_0.01,
       coef_Z_0.1 = coef_Z_0.1,
       coef_Z_0.8 = coef_Z_0.8,
       age_mean = age_mean,
       age_median = age_median,
       true.beta.in.CI_X = true.beta.in.CI_X,
       true.beta.in.CI_Z_0.0008 = true.beta.in.CI_Z_0.0008,
```

```
        true.beta.in.CI_Z_0.01 = true.beta.in.CI_Z_0.01,
        true.beta.in.CI_Z_0.1 = true.beta.in.CI_Z_0.1,
        true.beta.in.CI_Z_0.8 = true.beta.in.CI_Z_0.8)
  }

  else if (multiplicative == F) {
    list(coef_X = coef_X,
        coef_Z_5.36 = coef_Z_5.36,
        coef_Z_64.68 = coef_Z_64.68,
        coef_Z_794.31 = coef_Z_794.31,
        coef_Z_5023.42 = coef_Z_5023.42,
        age_mean = age_mean,
        age_median = age_median,
        true.beta.in.CI_X = true.beta.in.CI_X,
        true.beta.in.CI_Z_5.36 = true.beta.in.CI_Z_5.36,
        true.beta.in.CI_Z_64.68 = true.beta.in.CI_Z_64.68,
        true.beta.in.CI_Z_794.31 = true.beta.in.CI_Z_794.31,
        true.beta.in.CI_Z_5023.42 = true.beta.in.CI_Z_5023.42
    )

  }

  return(result)
}
```

## Data1

### Cluster

```
set.seed(12345)

data1_cluster <- cluster(data1)
data1_cluster[, Xshaft := mean(Xmean), by = c("mine", "shaft", "year")]

# Assignment error
data1_cluster[, `:=`(U_as_unshared_mu_0.077 = assignment_u(0.077, nrow(data1_cluster), T),
                     U_as_unshared_mu_0.1 = assignment_u(0.1, nrow(data1_cluster), T),
                     U_as_unshared_mu_0.8 = assignment_u(0.8, nrow(data1_cluster), T),
                     U_as_unshared_ad_190.18 = assignment_u(190.18, nrow(data1_cluster), F),
                     U_as_unshared_ad_250.15 = assignment_u(250.15, nrow(data1_cluster), F),
                     U_as_unshared_ad_2721.73 = assignment_u(2721.73, nrow(data1_cluster), F)
                     )
              ]

data1_cluster[, `:=`(U_as_shared_mu_0.077 = assignment_u(0.077, 1, T),
                     U_as_shared_mu_0.1 = assignment_u(0.1, 1, T),
                     U_as_shared_mu_0.8 = assignment_u(0.8, 1, T),
                     U_as_shared_ad_190.18 = assignment_u(190.18, 1, F),
                     U_as_shared_ad_250.15 = assignment_u(250.15, 1, F),
                     U_as_shared_ad_2721.73 = assignment_u(2721.73, 1, F)
                     ),
              by = ID]
```

```
data1_cluster[, `:=`(X_unshared_mu_0.077 = Xshaft * U_as_unshared_mu_0.077,
                     X_unshared_mu_0.1 = Xshaft * U_as_unshared_mu_0.1,
                     X_unshared_mu_0.8 = Xshaft * U_as_unshared_mu_0.8,
                     X_unshared_ad_190.18 = Xshaft + U_as_unshared_ad_190.18,
                     X_unshared_ad_250.15 = Xshaft + U_as_unshared_ad_250.15,
                     X_unshared_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73,
                     X_shared_mu_0.077 = Xshaft * U_as_shared_mu_0.077,
                     X_shared_mu_0.1 = Xshaft * U_as_shared_mu_0.1,
                     X_shared_mu_0.8 = Xshaft * U_as_shared_mu_0.8,
                     X_shared_ad_190.18 = Xshaft + U_as_shared_ad_190.18,
                     X_shared_ad_250.15 = Xshaft + U_as_shared_ad_250.15,
                     X_shared_ad_2721.73 = Xshaft + U_as_shared_ad_2721.73,
                     X_both_mu_0.077 = Xshaft * U_as_unshared_mu_0.077 * U_as_shared_mu_0.077,
                     X_both_mu_0.1 = Xshaft * U_as_unshared_mu_0.1 * U_as_shared_mu_0.1,
                     X_both_mu_0.8 = Xshaft * U_as_unshared_mu_0.8 * U_as_shared_mu_0.8,
                     X_both_ad_190.18 = Xshaft + U_as_unshared_ad_190.18 + U_as_shared_ad_190.18,
                     X_both_ad_250.15 = Xshaft + U_as_unshared_ad_250.15 + U_as_shared_ad_250.15,
                     X_both_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73 + U_as_shared_ad_2721.73
                     )
              ]


# Generalization Error


data1_cluster[, `:=`(U_ge_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                     U_ge_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                     U_ge_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                     U_ge_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                     U_ge_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                     U_ge_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                     U_ge_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                     U_ge_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                     ),
              by = c("mine", "shaft", "year")]


data1_cluster[, `:=`(Z_mu_0.0008 = Xshaft * U_ge_mu_0.0008,
                     Z_mu_0.01 = Xshaft * U_ge_mu_0.01,
                     Z_mu_0.1 = Xshaft * U_ge_mu_0.1,
                     Z_mu_0.8 = Xshaft * U_ge_mu_0.8,
                     Z_ad_5.36 = Xshaft + U_ge_ad_5.36,
                     Z_ad_64.68 = Xshaft + U_ge_ad_64.68,
                     Z_ad_794.31 = Xshaft + U_ge_ad_794.31,
                     Z_ad_5023.42 = Xshaft + U_ge_ad_5023.42
                     )
              ]

data1_cluster[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                     Zobject_mu_0.01 = mean(Z_mu_0.01),
                     Zobject_mu_0.1 = mean(Z_mu_0.1),
```

```
                      Zobject_mu_0.8 = mean(Z_mu_0.8),
                      Zobject_ad_5.36 = mean(Z_ad_5.36),
                      Zobject_ad_64.68 = mean(Z_ad_64.68),
                      Zobject_ad_794.31 = mean(Z_ad_794.31),
                      Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]



data1_cluster[, `:=`(Xcum_100_unshared_mu_0.077 = cumsum(X_unshared_mu_0.077)/100,
                      Xcum_100_unshared_mu_0.1 = cumsum(X_unshared_mu_0.1)/100,
                      Xcum_100_unshared_mu_0.8 = cumsum(X_unshared_mu_0.8)/100,
                      Xcum_100_unshared_ad_190.18 = cumsum(X_unshared_ad_190.18)/100,
                      Xcum_100_unshared_ad_250.15 = cumsum(X_unshared_ad_250.15)/100,
                      Xcum_100_unshared_ad_2721.73 = cumsum(X_unshared_ad_2721.73)/100,

                      Xcum_100_shared_mu_0.077 = cumsum(X_shared_mu_0.077)/100,
                      Xcum_100_shared_mu_0.1 = cumsum(X_shared_mu_0.1)/100,
                      Xcum_100_shared_mu_0.8 = cumsum(X_shared_mu_0.8)/100,
                      Xcum_100_shared_ad_190.18 = cumsum(X_shared_ad_190.18)/100,
                      Xcum_100_shared_ad_250.15 = cumsum(X_shared_ad_250.15)/100,
                      Xcum_100_shared_ad_2721.73 = cumsum(X_shared_ad_2721.73)/100,

                      Xcum_100_both_mu_0.077 = cumsum(X_both_mu_0.077)/100,
                      Xcum_100_both_mu_0.1 = cumsum(X_both_mu_0.1)/100,
                      Xcum_100_both_mu_0.8 = cumsum(X_both_mu_0.8)/100,
                      Xcum_100_both_ad_190.18 = cumsum(X_both_ad_190.18)/100,
                      Xcum_100_both_ad_250.15 = cumsum(X_both_ad_250.15)/100,
                      Xcum_100_both_ad_2721.73 = cumsum(X_both_ad_2721.73)/100,

                      Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                      Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                      Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                      Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                      Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                      Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                      Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                      Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                      ),
              by = ID]

cols <- c("ID", "age",
          names(data1_cluster)[stringr::str_detect(names(data1_cluster), "Zcum|Xcum")])
data1_cluster <- data1_cluster[,..cols]

for (i in base::intersect(names(data1_cluster)[stringr::str_detect(names(data1_cluster), "Xcum")],
                          names(data1_cluster)[stringr::str_detect(names(data1_cluster), "mu")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data1_assign_gener_delta_cluster_", Xcum),
         simulate_delta_assign_gener(data = data1_cluster, multiplicative = T,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data1_assign_gener_delta_cluster_", Xcum)),
```

```
          paste0("assign_gener/result_data1_assign_gener_delta_cluster_", Xcum, ".RDS"))

  assign(paste0("result_data1_assign_gener_delta_cluster_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data1_cluster, multiplicative = T,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data1_assign_gener_delta_cluster_", Xcum, "_censor")),
          paste0("assign_gener/result_data1_assign_gener_delta_cluster_", Xcum, "_censor.RDS"))

}

for (i in base::intersect(names(data1_cluster)[stringr::str_detect(names(data1_cluster), "Xcum")],
                          names(data1_cluster)[stringr::str_detect(names(data1_cluster), "ad")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data1_assign_gener_delta_cluster_", Xcum),
         simulate_delta_assign_gener(data = data1_cluster, multiplicative = F,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data1_assign_gener_delta_cluster_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_delta_cluster_", Xcum, ".RDS"))

  assign(paste0("result_data1_assign_gener_delta_cluster_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data1_cluster, multiplicative = F,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data1_assign_gener_delta_cluster_", Xcum, "_censor")),
          paste0("assign_gener/result_data1_assign_gener_delta_cluster_", Xcum, "_censor.RDS"))

}

for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data1_assign_gener_delta_cluster_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "mu")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 41, nchar(i) - 4)

  assign(paste0("result_data1_assign_gener_sim_cluster_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data1_assign_gener_delta_cluster_", Xcum)),
           multiplicative = T))
  saveRDS(get(paste0("result_data1_assign_gener_sim_cluster_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_sim_cluster_", Xcum, ".RDS"))
}

for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data1_assign_gener_delta_cluster_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "ad")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 41, nchar(i) - 4)

  assign(paste0("result_data1_assign_gener_sim_cluster_", Xcum),
```

```
        simulate_study_assign_gener(
          data = get(paste0("result_data1_assign_gener_delta_cluster_", Xcum)),
          multiplicative = F))
  saveRDS(get(paste0("result_data1_assign_gener_sim_cluster_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_sim_cluster_", Xcum, ".RDS"))
}
```

### Random

```
set.seed(12345)

data1_random <- data1[, `:=`(mine = sample(1:5, 1, replace = T),
                             shaft = sample(1:5, 1, replace = T)),
                      by = ID]
data1_random[, Xshaft := mean(X), by = c("mine", "shaft", "year")]

# Assignment error
data1_random[, `:=`(U_as_unshared_mu_0.077 = assignment_u(0.077, nrow(data1_random), T),
                    U_as_unshared_mu_0.1 = assignment_u(0.1, nrow(data1_random), T),
                    U_as_unshared_mu_0.8 = assignment_u(0.8, nrow(data1_random), T),
                    U_as_unshared_ad_190.18 = assignment_u(190.18, nrow(data1_random), F),
                    U_as_unshared_ad_250.15 = assignment_u(250.15, nrow(data1_random), F),
                    U_as_unshared_ad_2721.73 = assignment_u(2721.73, nrow(data1_random), F)
                    )
             ]

data1_random[, `:=`(U_as_shared_mu_0.077 = assignment_u(0.077, 1, T),
                    U_as_shared_mu_0.1 = assignment_u(0.1, 1, T),
                    U_as_shared_mu_0.8 = assignment_u(0.8, 1, T),
                    U_as_shared_ad_190.18 = assignment_u(190.18, 1, F),
                    U_as_shared_ad_250.15 = assignment_u(250.15, 1, F),
                    U_as_shared_ad_2721.73 = assignment_u(2721.73, 1, F)
                    ),
             by = ID]


data1_random[, `:=`(X_unshared_mu_0.077 = Xshaft * U_as_unshared_mu_0.077,
                    X_unshared_mu_0.1 = Xshaft * U_as_unshared_mu_0.1,
                    X_unshared_mu_0.8 = Xshaft * U_as_unshared_mu_0.8,
                    X_unshared_ad_190.18 = Xshaft + U_as_unshared_ad_190.18,
                    X_unshared_ad_250.15 = Xshaft + U_as_unshared_ad_250.15,
                    X_unshared_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73,
                    X_shared_mu_0.077 = Xshaft * U_as_shared_mu_0.077,
                    X_shared_mu_0.1 = Xshaft * U_as_shared_mu_0.1,
                    X_shared_mu_0.8 = Xshaft * U_as_shared_mu_0.8,
                    X_shared_ad_190.18 = Xshaft + U_as_shared_ad_190.18,
                    X_shared_ad_250.15 = Xshaft + U_as_shared_ad_250.15,
                    X_shared_ad_2721.73 = Xshaft + U_as_shared_ad_2721.73,
                    X_both_mu_0.077 = Xshaft * U_as_unshared_mu_0.077 * U_as_shared_mu_0.077,
                    X_both_mu_0.1 = Xshaft * U_as_unshared_mu_0.1 * U_as_shared_mu_0.1,
                    X_both_mu_0.8 = Xshaft * U_as_unshared_mu_0.8 * U_as_shared_mu_0.8,
                    X_both_ad_190.18 = Xshaft + U_as_unshared_ad_190.18 + U_as_shared_ad_190.18,
```

```
                    X_both_ad_250.15 = Xshaft + U_as_unshared_ad_250.15 + U_as_shared_ad_250.15,
                    X_both_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73 + U_as_shared_ad_2721.73
                    )
            ]


# Generalization Error


data1_random[, `:=`(U_ge_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                    U_ge_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                    U_ge_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                    U_ge_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                    U_ge_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                    U_ge_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                    U_ge_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                    U_ge_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                    ),
             by = c("mine", "shaft", "year")]


data1_random[, `:=`(Z_mu_0.0008 = Xshaft * U_ge_mu_0.0008,
                    Z_mu_0.01 = Xshaft * U_ge_mu_0.01,
                    Z_mu_0.1 = Xshaft * U_ge_mu_0.1,
                    Z_mu_0.8 = Xshaft * U_ge_mu_0.8,
                    Z_ad_5.36 = Xshaft + U_ge_ad_5.36,
                    Z_ad_64.68 = Xshaft + U_ge_ad_64.68,
                    Z_ad_794.31 = Xshaft + U_ge_ad_794.31,
                    Z_ad_5023.42 = Xshaft + U_ge_ad_5023.42
                    )
             ]

data1_random[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                    Zobject_mu_0.01 = mean(Z_mu_0.01),
                    Zobject_mu_0.1 = mean(Z_mu_0.1),
                    Zobject_mu_0.8 = mean(Z_mu_0.8),
                    Zobject_ad_5.36 = mean(Z_ad_5.36),
                    Zobject_ad_64.68 = mean(Z_ad_64.68),
                    Zobject_ad_794.31 = mean(Z_ad_794.31),
                    Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
             by = c("year", "mine")]



data1_random[, `:=`(Xcum_100_unshared_mu_0.077 = cumsum(X_unshared_mu_0.077)/100,
                    Xcum_100_unshared_mu_0.1 = cumsum(X_unshared_mu_0.1)/100,
                    Xcum_100_unshared_mu_0.8 = cumsum(X_unshared_mu_0.8)/100,
                    Xcum_100_unshared_ad_190.18 = cumsum(X_unshared_ad_190.18)/100,
                    Xcum_100_unshared_ad_250.15 = cumsum(X_unshared_ad_250.15)/100,
                    Xcum_100_unshared_ad_2721.73 = cumsum(X_unshared_ad_2721.73)/100,

                    Xcum_100_shared_mu_0.077 = cumsum(X_shared_mu_0.077)/100,
                    Xcum_100_shared_mu_0.1 = cumsum(X_shared_mu_0.1)/100,
```

```
                    Xcum_100_shared_mu_0.8 = cumsum(X_shared_mu_0.8)/100,
                    Xcum_100_shared_ad_190.18 = cumsum(X_shared_ad_190.18)/100,
                    Xcum_100_shared_ad_250.15 = cumsum(X_shared_ad_250.15)/100,
                    Xcum_100_shared_ad_2721.73 = cumsum(X_shared_ad_2721.73)/100,

                    Xcum_100_both_mu_0.077 = cumsum(X_both_mu_0.077)/100,
                    Xcum_100_both_mu_0.1 = cumsum(X_both_mu_0.1)/100,
                    Xcum_100_both_mu_0.8 = cumsum(X_both_mu_0.8)/100,
                    Xcum_100_both_ad_190.18 = cumsum(X_both_ad_190.18)/100,
                    Xcum_100_both_ad_250.15 = cumsum(X_both_ad_250.15)/100,
                    Xcum_100_both_ad_2721.73 = cumsum(X_both_ad_2721.73)/100,

                    Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                    Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                    Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                    Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                    Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                    Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                    Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                    Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                    ),
                by = ID]

cols <- c("ID", "age",
          names(data1_random)[stringr::str_detect(names(data1_random), "Zcum|Xcum")])
data1_random <- data1_random[,..cols]


for (i in base::intersect(names(data1_random)[stringr::str_detect(names(data1_random), "Xcum")],
                          names(data1_random)[stringr::str_detect(names(data1_random), "mu")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data1_assign_gener_delta_random_", Xcum),
         simulate_delta_assign_gener(data = data1_random, multiplicative = T,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data1_assign_gener_delta_random_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_delta_random_", Xcum, ".RDS"))

  assign(paste0("result_data1_assign_gener_delta_random_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data1_random, multiplicative = T,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data1_assign_gener_delta_random_", Xcum, "_censor")),
          paste0("assign_gener/result_data1_assign_gener_delta_random_", Xcum, "_censor.RDS"))

}

for (i in base::intersect(names(data1_random)[stringr::str_detect(names(data1_random), "Xcum")],
                          names(data1_random)[stringr::str_detect(names(data1_random), "ad")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data1_assign_gener_delta_random_", Xcum),
         simulate_delta_assign_gener(data = data1_random, multiplicative = F,
                                     Xcum_100 = paste0(i)))
```

```
  saveRDS(get(paste0("result_data1_assign_gener_delta_random_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_delta_random_", Xcum, ".RDS"))

  assign(paste0("result_data1_assign_gener_delta_random_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data1_random, multiplicative = F,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data1_assign_gener_delta_random_", Xcum, "_censor")),
          paste0("assign_gener/result_data1_assign_gener_delta_random_", Xcum, "_censor.RDS"))

}


for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data1_assign_gener_delta_random_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "mu")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 40, nchar(i) - 4)

  assign(paste0("result_data1_assign_gener_sim_random_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data1_assign_gener_delta_random_", Xcum)),
           multiplicative = T))
  saveRDS(get(paste0("result_data1_assign_gener_sim_random_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_sim_random_", Xcum, ".RDS"))
}

for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data1_assign_gener_delta_random_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "ad")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 40, nchar(i) - 4)

  assign(paste0("result_data1_assign_gener_sim_random_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data1_assign_gener_delta_random_", Xcum)),
           multiplicative = F))
  saveRDS(get(paste0("result_data1_assign_gener_sim_random_", Xcum)),
          paste0("assign_gener/result_data1_assign_gener_sim_random_", Xcum, ".RDS"))
}
```

## Data2

### Cluster

```
set.seed(12345)

data2_cluster <- cluster(data2)
data2_cluster[, Xshaft := mean(Xmean), by = c("mine", "shaft", "year")]
```

```r
# Assignment error
data2_cluster[, `:=`(U_as_unshared_mu_0.077 = assignment_u(0.077, nrow(data2_cluster), T),
                     U_as_unshared_mu_0.1 = assignment_u(0.1, nrow(data2_cluster), T),
                     U_as_unshared_mu_0.8 = assignment_u(0.8, nrow(data2_cluster), T),
                     U_as_unshared_ad_190.18 = assignment_u(190.18, nrow(data2_cluster), F),
                     U_as_unshared_ad_250.15 = assignment_u(250.15, nrow(data2_cluster), F),
                     U_as_unshared_ad_2721.73 = assignment_u(2721.73, nrow(data2_cluster), F)
                     )
              ]

data2_cluster[, `:=`(U_as_shared_mu_0.077 = assignment_u(0.077, 1, T),
                     U_as_shared_mu_0.1 = assignment_u(0.1, 1, T),
                     U_as_shared_mu_0.8 = assignment_u(0.8, 1, T),
                     U_as_shared_ad_190.18 = assignment_u(190.18, 1, F),
                     U_as_shared_ad_250.15 = assignment_u(250.15, 1, F),
                     U_as_shared_ad_2721.73 = assignment_u(2721.73, 1, F)
                     ),
              by = ID]


data2_cluster[, `:=`(X_unshared_mu_0.077 = Xshaft * U_as_unshared_mu_0.077,
                     X_unshared_mu_0.1 = Xshaft * U_as_unshared_mu_0.1,
                     X_unshared_mu_0.8 = Xshaft * U_as_unshared_mu_0.8,
                     X_unshared_ad_190.18 = Xshaft + U_as_unshared_ad_190.18,
                     X_unshared_ad_250.15 = Xshaft + U_as_unshared_ad_250.15,
                     X_unshared_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73,
                     X_shared_mu_0.077 = Xshaft * U_as_shared_mu_0.077,
                     X_shared_mu_0.1 = Xshaft * U_as_shared_mu_0.1,
                     X_shared_mu_0.8 = Xshaft * U_as_shared_mu_0.8,
                     X_shared_ad_190.18 = Xshaft + U_as_shared_ad_190.18,
                     X_shared_ad_250.15 = Xshaft + U_as_shared_ad_250.15,
                     X_shared_ad_2721.73 = Xshaft + U_as_shared_ad_2721.73,
                     X_both_mu_0.077 = Xshaft * U_as_unshared_mu_0.077 * U_as_shared_mu_0.077,
                     X_both_mu_0.1 = Xshaft * U_as_unshared_mu_0.1 * U_as_shared_mu_0.1,
                     X_both_mu_0.8 = Xshaft * U_as_unshared_mu_0.8 * U_as_shared_mu_0.8,
                     X_both_ad_190.18 = Xshaft + U_as_unshared_ad_190.18 + U_as_shared_ad_190.18,
                     X_both_ad_250.15 = Xshaft + U_as_unshared_ad_250.15 + U_as_shared_ad_250.15,
                     X_both_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73 + U_as_shared_ad_2721.73
                     )
              ]


# Generalization Error


data2_cluster[, `:=`(U_ge_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                     U_ge_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                     U_ge_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                     U_ge_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                     U_ge_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                     U_ge_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                     U_ge_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                     U_ge_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
```

```
                    ),
              by = c("mine", "shaft", "year")]


data2_cluster[, `:=`(Z_mu_0.0008 = Xshaft * U_ge_mu_0.0008,
                     Z_mu_0.01 = Xshaft * U_ge_mu_0.01,
                     Z_mu_0.1 = Xshaft * U_ge_mu_0.1,
                     Z_mu_0.8 = Xshaft * U_ge_mu_0.8,
                     Z_ad_5.36 = Xshaft + U_ge_ad_5.36,
                     Z_ad_64.68 = Xshaft + U_ge_ad_64.68,
                     Z_ad_794.31 = Xshaft + U_ge_ad_794.31,
                     Z_ad_5023.42 = Xshaft + U_ge_ad_5023.42
                     )
              ]

data2_cluster[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                     Zobject_mu_0.01 = mean(Z_mu_0.01),
                     Zobject_mu_0.1 = mean(Z_mu_0.1),
                     Zobject_mu_0.8 = mean(Z_mu_0.8),
                     Zobject_ad_5.36 = mean(Z_ad_5.36),
                     Zobject_ad_64.68 = mean(Z_ad_64.68),
                     Zobject_ad_794.31 = mean(Z_ad_794.31),
                     Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]



data2_cluster[, `:=`(Xcum_100_unshared_mu_0.077 = cumsum(X_unshared_mu_0.077)/100,
                     Xcum_100_unshared_mu_0.1 = cumsum(X_unshared_mu_0.1)/100,
                     Xcum_100_unshared_mu_0.8 = cumsum(X_unshared_mu_0.8)/100,
                     Xcum_100_unshared_ad_190.18 = cumsum(X_unshared_ad_190.18)/100,
                     Xcum_100_unshared_ad_250.15 = cumsum(X_unshared_ad_250.15)/100,
                     Xcum_100_unshared_ad_2721.73 = cumsum(X_unshared_ad_2721.73)/100,

                     Xcum_100_shared_mu_0.077 = cumsum(X_shared_mu_0.077)/100,
                     Xcum_100_shared_mu_0.1 = cumsum(X_shared_mu_0.1)/100,
                     Xcum_100_shared_mu_0.8 = cumsum(X_shared_mu_0.8)/100,
                     Xcum_100_shared_ad_190.18 = cumsum(X_shared_ad_190.18)/100,
                     Xcum_100_shared_ad_250.15 = cumsum(X_shared_ad_250.15)/100,
                     Xcum_100_shared_ad_2721.73 = cumsum(X_shared_ad_2721.73)/100,

                     Xcum_100_both_mu_0.077 = cumsum(X_both_mu_0.077)/100,
                     Xcum_100_both_mu_0.1 = cumsum(X_both_mu_0.1)/100,
                     Xcum_100_both_mu_0.8 = cumsum(X_both_mu_0.8)/100,
                     Xcum_100_both_ad_190.18 = cumsum(X_both_ad_190.18)/100,
                     Xcum_100_both_ad_250.15 = cumsum(X_both_ad_250.15)/100,
                     Xcum_100_both_ad_2721.73 = cumsum(X_both_ad_2721.73)/100,

                     Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                     Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                     Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                     Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                     Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
```

```
                    Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                    Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                    Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                    ),
              by = ID]

cols <- c("ID", "age", names(data2_cluster)[stringr::str_detect(names(data2_cluster), "Zcum|Xcum")])
data2_cluster <- data2_cluster[,..cols]

for (i in base::intersect(names(data2_cluster)[stringr::str_detect(names(data2_cluster), "Xcum")],
                          names(data2_cluster)[stringr::str_detect(names(data2_cluster), "mu")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data2_assign_gener_delta_cluster_", Xcum),
         simulate_delta_assign_gener(data = data2_cluster, multiplicative = T,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data2_assign_gener_delta_cluster_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_delta_cluster_", Xcum, ".RDS"))

  assign(paste0("result_data2_assign_gener_delta_cluster_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data2_cluster, multiplicative = T,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data2_assign_gener_delta_cluster_", Xcum, "_censor")),
          paste0("assign_gener/result_data2_assign_gener_delta_cluster_", Xcum, "_censor.RDS"))

}

for (i in base::intersect(names(data2_cluster)[stringr::str_detect(names(data2_cluster), "Xcum")],
                          names(data2_cluster)[stringr::str_detect(names(data2_cluster), "ad")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data2_assign_gener_delta_cluster_", Xcum),
         simulate_delta_assign_gener(data = data2_cluster, multiplicative = F,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data2_assign_gener_delta_cluster_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_delta_cluster_", Xcum, ".RDS"))

  assign(paste0("result_data2_assign_gener_delta_cluster_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data2_cluster, multiplicative = F,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data2_assign_gener_delta_cluster_", Xcum, "_censor")),
          paste0("assign_gener/result_data2_assign_gener_delta_cluster_", Xcum, "_censor.RDS"))

}




for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data2_assign_gener_delta_cluster_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "mu")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))
```

```
  Xcum <- stringr::str_sub(i, 41, nchar(i) - 4)

  assign(paste0("result_data2_assign_gener_sim_cluster_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data2_assign_gener_delta_cluster_", Xcum)),
           multiplicative = T))
  saveRDS(get(paste0("result_data2_assign_gener_sim_cluster_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_sim_cluster_", Xcum, ".RDS"))
}

for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data2_assign_gener_delta_cluster_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "ad")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 41, nchar(i) - 4)

  assign(paste0("result_data2_assign_gener_sim_cluster_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data2_assign_gener_delta_cluster_", Xcum)),
           multiplicative = F))
  saveRDS(get(paste0("result_data2_assign_gener_sim_cluster_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_sim_cluster_", Xcum, ".RDS"))
}
```

### Random

```
set.seed(12345)

data2_random <- data2[, `:=`(mine = sample(1:5, 1, replace = T),
                             shaft = sample(1:5, 1, replace = T)),
                      by = ID]
data2_random[, Xshaft := mean(X), by = c("mine", "shaft", "year")]

# Assignment error
data2_random[, `:=`(U_as_unshared_mu_0.077 = assignment_u(0.077, nrow(data2_random), T),
                    U_as_unshared_mu_0.1 = assignment_u(0.1, nrow(data2_random), T),
                    U_as_unshared_mu_0.8 = assignment_u(0.8, nrow(data2_random), T),
                    U_as_unshared_ad_190.18 = assignment_u(190.18, nrow(data2_random), F),
                    U_as_unshared_ad_250.15 = assignment_u(250.15, nrow(data2_random), F),
                    U_as_unshared_ad_2721.73 = assignment_u(2721.73, nrow(data2_random), F)
                    )
             ]

data2_random[, `:=`(U_as_shared_mu_0.077 = assignment_u(0.077, 1, T),
                    U_as_shared_mu_0.1 = assignment_u(0.1, 1, T),
                    U_as_shared_mu_0.8 = assignment_u(0.8, 1, T),
                    U_as_shared_ad_190.18 = assignment_u(190.18, 1, F),
                    U_as_shared_ad_250.15 = assignment_u(250.15, 1, F),
                    U_as_shared_ad_2721.73 = assignment_u(2721.73, 1, F)
                    ),
```

```
             by = ID]


data2_random[, `:=`(X_unshared_mu_0.077 = Xshaft * U_as_unshared_mu_0.077,
                    X_unshared_mu_0.1 = Xshaft * U_as_unshared_mu_0.1,
                    X_unshared_mu_0.8 = Xshaft * U_as_unshared_mu_0.8,
                    X_unshared_ad_190.18 = Xshaft + U_as_unshared_ad_190.18,
                    X_unshared_ad_250.15 = Xshaft + U_as_unshared_ad_250.15,
                    X_unshared_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73,
                    X_shared_mu_0.077 = Xshaft * U_as_shared_mu_0.077,
                    X_shared_mu_0.1 = Xshaft * U_as_shared_mu_0.1,
                    X_shared_mu_0.8 = Xshaft * U_as_shared_mu_0.8,
                    X_shared_ad_190.18 = Xshaft + U_as_shared_ad_190.18,
                    X_shared_ad_250.15 = Xshaft + U_as_shared_ad_250.15,
                    X_shared_ad_2721.73 = Xshaft + U_as_shared_ad_2721.73,
                    X_both_mu_0.077 = Xshaft * U_as_unshared_mu_0.077 * U_as_shared_mu_0.077,
                    X_both_mu_0.1 = Xshaft * U_as_unshared_mu_0.1 * U_as_shared_mu_0.1,
                    X_both_mu_0.8 = Xshaft * U_as_unshared_mu_0.8 * U_as_shared_mu_0.8,
                    X_both_ad_190.18 = Xshaft + U_as_unshared_ad_190.18 + U_as_shared_ad_190.18,
                    X_both_ad_250.15 = Xshaft + U_as_unshared_ad_250.15 + U_as_shared_ad_250.15,
                    X_both_ad_2721.73 = Xshaft + U_as_unshared_ad_2721.73 + U_as_shared_ad_2721.73
                    )
             ]

# Generalization Error


data2_random[, `:=`(U_ge_mu_0.0008 = assignment_u(var = 0.0008, n = 1, multiplicative = T),
                    U_ge_mu_0.01 = assignment_u(var = 0.01, n = 1, multiplicative = T),
                    U_ge_mu_0.1 = assignment_u(var = 0.1, n = 1, multiplicative = T),
                    U_ge_mu_0.8 = assignment_u(var = 0.8, n = 1, multiplicative = T),
                    U_ge_ad_5.36 = assignment_u(var = 5.36, n = 1, multiplicative = F),
                    U_ge_ad_64.68 = assignment_u(var = 64.68, n = 1, multiplicative = F),
                    U_ge_ad_794.31 = assignment_u(var = 794.31, n = 1, multiplicative = F),
                    U_ge_ad_5023.42 = assignment_u(var = 5023.42, n = 1, multiplicative = F)
                    ),
              by = c("mine", "shaft", "year")]


data2_random[, `:=`(Z_mu_0.0008 = Xshaft * U_ge_mu_0.0008,
                    Z_mu_0.01 = Xshaft * U_ge_mu_0.01,
                    Z_mu_0.1 = Xshaft * U_ge_mu_0.1,
                    Z_mu_0.8 = Xshaft * U_ge_mu_0.8,
                    Z_ad_5.36 = Xshaft + U_ge_ad_5.36,
                    Z_ad_64.68 = Xshaft + U_ge_ad_64.68,
                    Z_ad_794.31 = Xshaft + U_ge_ad_794.31,
                    Z_ad_5023.42 = Xshaft + U_ge_ad_5023.42
                    )
              ]

data2_random[, `:=`(Zobject_mu_0.0008 = mean(Z_mu_0.0008),
                    Zobject_mu_0.01 = mean(Z_mu_0.01),
                    Zobject_mu_0.1 = mean(Z_mu_0.1),
```

```
                      Zobject_mu_0.8 = mean(Z_mu_0.8),
                      Zobject_ad_5.36 = mean(Z_ad_5.36),
                      Zobject_ad_64.68 = mean(Z_ad_64.68),
                      Zobject_ad_794.31 = mean(Z_ad_794.31),
                      Zobject_ad_5023.42 = mean(Z_ad_5023.42)),
              by = c("year", "mine")]



data2_random[, `:=`(Xcum_100_unshared_mu_0.077 = cumsum(X_unshared_mu_0.077)/100,
                      Xcum_100_unshared_mu_0.1 = cumsum(X_unshared_mu_0.1)/100,
                      Xcum_100_unshared_mu_0.8 = cumsum(X_unshared_mu_0.8)/100,
                      Xcum_100_unshared_ad_190.18 = cumsum(X_unshared_ad_190.18)/100,
                      Xcum_100_unshared_ad_250.15 = cumsum(X_unshared_ad_250.15)/100,
                      Xcum_100_unshared_ad_2721.73 = cumsum(X_unshared_ad_2721.73)/100,

                      Xcum_100_shared_mu_0.077 = cumsum(X_shared_mu_0.077)/100,
                      Xcum_100_shared_mu_0.1 = cumsum(X_shared_mu_0.1)/100,
                      Xcum_100_shared_mu_0.8 = cumsum(X_shared_mu_0.8)/100,
                      Xcum_100_shared_ad_190.18 = cumsum(X_shared_ad_190.18)/100,
                      Xcum_100_shared_ad_250.15 = cumsum(X_shared_ad_250.15)/100,
                      Xcum_100_shared_ad_2721.73 = cumsum(X_shared_ad_2721.73)/100,

                      Xcum_100_both_mu_0.077 = cumsum(X_both_mu_0.077)/100,
                      Xcum_100_both_mu_0.1 = cumsum(X_both_mu_0.1)/100,
                      Xcum_100_both_mu_0.8 = cumsum(X_both_mu_0.8)/100,
                      Xcum_100_both_ad_190.18 = cumsum(X_both_ad_190.18)/100,
                      Xcum_100_both_ad_250.15 = cumsum(X_both_ad_250.15)/100,
                      Xcum_100_both_ad_2721.73 = cumsum(X_both_ad_2721.73)/100,

                      Zcum_100_object_mu_0.0008 = cumsum(Zobject_mu_0.0008)/100,
                      Zcum_100_object_mu_0.01 = cumsum(Zobject_mu_0.01)/100,
                      Zcum_100_object_mu_0.1 = cumsum(Zobject_mu_0.1)/100,
                      Zcum_100_object_mu_0.8 = cumsum(Zobject_mu_0.8)/100,
                      Zcum_100_object_ad_5.36 = cumsum(Zobject_ad_5.36)/100,
                      Zcum_100_object_ad_64.68 = cumsum(Zobject_ad_64.68)/100,
                      Zcum_100_object_ad_794.31 = cumsum(Zobject_ad_794.31)/100,
                      Zcum_100_object_ad_5023.42 = cumsum(Zobject_ad_5023.42)/100
                      ),
              by = ID]

cols <- c("ID", "age",
          names(data2_random)[stringr::str_detect(names(data2_random), "Zcum|Xcum")])
data2_random <- data2_random[,..cols]


for (i in base::intersect(names(data2_random)[stringr::str_detect(names(data2_random), "Xcum")],
                          names(data2_random)[stringr::str_detect(names(data2_random), "mu")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data2_assign_gener_delta_random_", Xcum),
         simulate_delta_assign_gener(data = data2_random, multiplicative = T,
                                     Xcum_100 = paste0(i)))
```

```
  saveRDS(get(paste0("result_data2_assign_gener_delta_random_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_delta_random_", Xcum, ".RDS"))

  assign(paste0("result_data2_assign_gener_delta_random_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data2_random, multiplicative = T,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data2_assign_gener_delta_random_", Xcum, "_censor")),
          paste0("assign_gener/result_data2_assign_gener_delta_random_", Xcum, "_censor.RDS"))

}

for (i in base::intersect(names(data2_random)[stringr::str_detect(names(data2_random), "Xcum")],
                          names(data2_random)[stringr::str_detect(names(data2_random), "ad")])) {
  Xcum <- stringr::str_sub(i, 10)

  assign(paste0("result_data2_assign_gener_delta_random_", Xcum),
         simulate_delta_assign_gener(data = data2_random, multiplicative = F,
                                     Xcum_100 = paste0(i)))
  saveRDS(get(paste0("result_data2_assign_gener_delta_random_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_delta_random_", Xcum, ".RDS"))

  assign(paste0("result_data2_assign_gener_delta_random_", Xcum, "_censor"),
         simulate_delta_assign_gener(data = data2_random, multiplicative = F,
                                     Xcum_100 = paste0(i), censor = 0.1))
  saveRDS(get(paste0("result_data2_assign_gener_delta_random_", Xcum, "_censor")),
          paste0("assign_gener/result_data2_assign_gener_delta_random_", Xcum, "_censor.RDS"))

}


for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data2_assign_gener_delta_random_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "mu")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 40, nchar(i) - 4)

  assign(paste0("result_data2_assign_gener_sim_random_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data2_assign_gener_delta_random_", Xcum)),
           multiplicative = T))
  saveRDS(get(paste0("result_data2_assign_gener_sim_random_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_sim_random_", Xcum, ".RDS"))
}

for (i in base::intersect(list.files("assign_gener")[stringr::str_detect(
  list.files("assign_gener"), "data2_assign_gener_delta_random_")],
  list.files("assign_gener")[stringr::str_detect(list.files("assign_gener"), "ad")])) {

  assign(substr(i, 1, nchar(i) - 4), readRDS(paste0("assign_gener/", i)))

  Xcum <- stringr::str_sub(i, 40, nchar(i) - 4)
```

```
  assign(paste0("result_data2_assign_gener_sim_random_", Xcum),
         simulate_study_assign_gener(
           data = get(paste0("result_data2_assign_gener_delta_random_", Xcum)),
           multiplicative = F))
  saveRDS(get(paste0("result_data2_assign_gener_sim_random_", Xcum)),
          paste0("assign_gener/result_data2_assign_gener_sim_random_", Xcum, ".RDS"))
}
```

## Estimation error

```
set.seed(12345)

CRn_density <- density(
  c(1100, 700, 4070, 16600, 8900, 2800, 590, 2600, 19000, 1800, 3600, 1370)/1000,
  from = 0, to = 19, n = 1000)

CRn <- data.table(CRn = CRn_density$x, density = CRn_density$y)
dens_sum <- sum(CRn$density)
CRn[, p_appr := density/dens_sum]

activity_weighting_factors <- c(0,
                                rep(0.1, 4),
                                rep(0.2, 19),
                                rep(0.3, 3),
                                rep(0.4, 6),
                                rep(0.5, 10),
                                rep(0.6, 9),
                                rep(0.8, 3),
                                rep(1, 6))

working_time_factors <- c(rep(0.88, length(46:58)),
                          rep(0.9,  length(65)),
                          rep(1.1, length(66:80)),
                          rep(1.2, length(81:90)))

equilibrium_factors <- c(0.2,
                         rep(0.3, 44),
                         rep(0.4, 18),
                         rep(0.5, 22),
                         rep(0.6, 9))

correction_factors <- c(1.2, rep(1.3, 5), 1.4, 1.45)
```

## Functions

### Generating time of death

Simulate.Survival.Time_estim() creates the time of death for one miner with an estimation error. It is an adapted variant of Hendry's function.

```
Simulate.Survival.Time_estim <-
  function(time.interval, true.X.cum, obs.Z.cum, beta, basehaz, censor = NULL) {

    # CREATING g() AND g^-1()
    g <- function(x) {
      x / basehaz
    }
    g.inv <- function(x) {
      x * basehaz
    }
```

```r
#CREATING THE BOUNDS OF TRUNCATION
t.max <- 100
t.min <- time.interval[1]

g.inv.t.max <- g.inv(t.max)
g.inv.t.min <- g.inv(t.min)

time.interval <- c(0, time.interval, 110)
g.inv.t <- g.inv(time.interval)

X.cum <- c(0, true.X.cum, true.X.cum[length(true.X.cum)])

obs.Z.cum[[1]] <- c(obs.Z.cum[[1]], obs.Z.cum[[1]][length(obs.Z.cum)])
obs.Z.cum[[2]] <- c(obs.Z.cum[[2]], obs.Z.cum[[2]][length(obs.Z.cum)])
obs.Z.cum[[3]] <- c(obs.Z.cum[[3]], obs.Z.cum[[3]][length(obs.Z.cum)])
obs.Z.cum[[4]] <- c(obs.Z.cum[[4]], obs.Z.cum[[4]][length(obs.Z.cum)])
obs.Z.cum[[5]] <- c(obs.Z.cum[[5]], obs.Z.cum[[5]][length(obs.Z.cum)])
obs.Z.cum[[6]] <- c(obs.Z.cum[[6]], obs.Z.cum[[6]][length(obs.Z.cum)])
obs.Z.cum[[7]] <- c(obs.Z.cum[[7]], obs.Z.cum[[7]][length(obs.Z.cum)])
obs.Z.cum[[8]] <- c(obs.Z.cum[[8]], obs.Z.cum[[8]][length(obs.Z.cum)])
obs.Z.cum[[9]] <- c(obs.Z.cum[[9]], obs.Z.cum[[9]][length(obs.Z.cum)])
obs.Z.cum[[10]] <- c(obs.Z.cum[[10]], obs.Z.cum[[10]][length(obs.Z.cum)])
obs.Z.cum[[11]] <- c(obs.Z.cum[[11]], obs.Z.cum[[11]][length(obs.Z.cum)])
obs.Z.cum[[12]] <- c(obs.Z.cum[[12]], obs.Z.cum[[12]][length(obs.Z.cum)])

lambda <- exp(beta * X.cum)

#GENERATING DATA USING ACCEPT-REJECT METHOD
k <- function(x, m, M, rates, t) {
  ifelse(x <= m | x >= M, 0, dpexp(x, rates, t))
}

gen.y <- function(x) {
  r <- 60
  repeat {
    y <- rpexp(r, x, g.inv.t)
    u <- runif(r)
    t <-
      (k(y, g.inv.t.min, g.inv.t.max, x, g.inv.t)) / (dpexp(y, x, g.inv.t))
    y <- y[u <= t][1]
    if (!is.na(y))
      break
  }
  y
}

g.y <-  g(gen.y(lambda))

#CREATING CENSORING INDICATOR
if (is.null(censor)) {
  Y = g.y
} else{
  C <- rexp(1, rate = censor)
```

```
  while (C <= t.min) {
    C <- rexp(1, rate = censor)
  }
  Y = min(C, g.y)
}
#CREATING DATASET
data <- data.table(
  time.interval[-c(1, which(time.interval >= Y))],
  c(time.interval[-c(1, 2, which(time.interval >= Y))], Y),
  X.cum[2:(length(time.interval[-c(1, which(time.interval >= Y))]) + 1)],
  obs.Z.cum[[1]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[2]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[3]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[4]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[5]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[6]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[7]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[8]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[9]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[10]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[11]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  obs.Z.cum[[12]][1:length(time.interval[-c(1, which(time.interval >= Y))])],
  rep(0, length(time.interval[-c(1, which(time.interval >= Y))]))
  )

colnames(data) <-
  c("start",
    "stop",
    "true.cum.exposure",
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[1]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[2]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[3]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[4]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[5]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[6]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[7]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[8]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[9]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[10]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[11]], 10, 30)),
    paste0("obs.cum.exposure_", substr(names(obs.Z.cum)[[12]], 10, 30)),
    "delta")

if (is.null(censor)) {
  data$delta[dim(data)[1]] <- 1
} else if (C > g.y) {
  data$delta[dim(data)[1]] <- 1
}


# Against error: Fehler in aeqSurv(Y) :
# aeqSurv exception, an interval has effective length 0
data[stop - start < 0.001, stop := stop + 0.001]
```

```
    return(data)

}
```

simulate_delta_estim() randomly selects 500 miners and calculates their time of death. The entire process is repeated 500 times.

```
simulate_delta_estim <- function(data = data, n_rep = 500, n_miners = 500,
                                 beta = 0.005, basehaz = 0.05, censor = NULL) {

  set.seed(12345)
  seeds <- sample(1:99999, n_rep)

  results <- list()

  cols <- names(data)[stringr::str_detect(names(data), "Zcum_100")]

  for (i in 1:n_rep) {
    set.seed(seeds[i])
    results[[i]] <-
      foreach(j = sample(unique(data$ID), n_miners), .combine = rbind) %do% {

        obs.Z.cum <- list(
          data[ID == j,][[cols[[1]]]],
          data[ID == j,][[cols[[2]]]],
          data[ID == j,][[cols[[3]]]],
          data[ID == j,][[cols[[4]]]],
          data[ID == j,][[cols[[5]]]],
          data[ID == j,][[cols[[6]]]],
          data[ID == j,][[cols[[7]]]],
          data[ID == j,][[cols[[8]]]],
          data[ID == j,][[cols[[9]]]],
          data[ID == j,][[cols[[10]]]],
          data[ID == j,][[cols[[11]]]],
          data[ID == j,][[cols[[12]]]])

        names(obs.Z.cum) <- cols

      Simulate.Survival.Time_estim(
        time.interval = data[ID == j,]$age,
        true.X.cum = data[ID == j,]$Xcum_100,
        obs.Z.cum = obs.Z.cum,
        beta = beta,
        basehaz = basehaz,
        censor = censor)
      }

  }
  return(results)
}
```

simulate_study_estim() fits Cox models and returns various values of interest.

```r
simulate_study_estim <- function(data = simulate_delta_estim, beta = 0.005, C) {

  true_beta <- beta
  coef_X <- c()

  for (i in names(data[[1]])[stringr::str_detect(names(data[[1]]), "obs.cum.exposure_")]) {
    assign(paste0("coef_Z_", substr(i, 18, 50)), c())
    assign(paste0("true.beta.in.CI_Z_", substr(i, 18, 50)), rep("No", length(data)))
  }

  age_mean <- c()
  age_median <- c()

  true.beta.in.CI_X <- rep("No", length(data))

  for (i in 1:length(data)) {
    setDT(data[[i]])
    model_X <- coxph(Surv(start, stop, delta) ~ true.cum.exposure,
                     data = data[[i]], control = coxph.control(timefix = FALSE))

    for (j in names(data[[i]])[stringr::str_detect(names(data[[i]]), "obs.cum.exposure_")]) {

      assign(paste0("model_Z_", substr(j, 18, 50)),
             coxph(Surv(start, stop, delta) ~ get(j),
                   data = data[[i]], control = coxph.control(timefix = FALSE)))

      assign(paste0("coef_Z_", substr(j, 18, 50)),
             c(get(paste0("coef_Z_", substr(j, 18, 50))),
               get(paste0("model_Z_", substr(j, 18, 50)))$coefficients))

      assign(paste0("lower_Z_", substr(j, 18, 50)),
             coef(get(paste0("model_Z_", substr(j, 18, 50)))) -
               qnorm(0.975) * sqrt(get(paste0("model_Z_", substr(j, 18, 50)))$var))

      assign(paste0("upper_Z_", substr(j, 18, 50)),
             coef(get(paste0("model_Z_", substr(j, 18, 50)))) +
               qnorm(0.975) * sqrt(get(paste0("model_Z_", substr(j, 18, 50)))$var))

      }



    coef_X <- c(coef_X, model_X$coefficients)

    age_mean <- c(age_mean, mean(data[[i]][delta == 1,]$stop))
    age_median <- c(age_median, median(data[[i]][delta == 1,]$stop))

    lower_X <- coef(model_X) - qnorm(0.975) * sqrt(model_X$var)
    upper_X <- coef(model_X) + qnorm(0.975) * sqrt(model_X$var)
    if (lower_X < true_beta & true_beta < upper_X) {
      true.beta.in.CI_X[i] <- "yes"
    }
```

```
if (lower_Z_w_unshared_mu < true_beta & true_beta < upper_Z_w_unshared_mu) {
  true.beta.in.CI_Z_w_unshared_mu[i] <- "yes"
}
if (lower_Z_w_unshared_ad < true_beta & true_beta < upper_Z_w_unshared_ad) {
  true.beta.in.CI_Z_w_unshared_ad[i] <- "yes"
}

if (C == "CRn") {
  if (lower_Z_g_unshared_mu < true_beta & true_beta < upper_Z_g_unshared_mu) {
    true.beta.in.CI_Z_g_unshared_mu[i] <- "yes"
  }
  if (lower_Z_g_unshared_ad < true_beta & true_beta < upper_Z_g_unshared_ad) {
    true.beta.in.CI_Z_g_unshared_ad[i] <- "yes"
  }
}

if (C == "CRDP") {
  if (lower_Z_c_unshared_mu < true_beta & true_beta < upper_Z_c_unshared_mu) {
    true.beta.in.CI_Z_c_unshared_mu[i] <- "yes"
  }
  if (lower_Z_c_unshared_ad < true_beta & true_beta < upper_Z_c_unshared_ad) {
    true.beta.in.CI_Z_c_unshared_ad[i] <- "yes"
  }
}

if (lower_Z_f_unshared_mu < true_beta & true_beta < upper_Z_f_unshared_mu) {
  true.beta.in.CI_Z_f_unshared_mu[i] <- "yes"
}
if (lower_Z_f_unshared_ad < true_beta & true_beta < upper_Z_f_unshared_ad) {
  true.beta.in.CI_Z_f_unshared_ad[i] <- "yes"
}
if (lower_Z_w_shared_mu < true_beta & true_beta < upper_Z_w_shared_mu) {
  true.beta.in.CI_Z_w_shared_mu[i] <- "yes"
}
if (lower_Z_w_shared_ad < true_beta & true_beta < upper_Z_w_shared_ad) {
  true.beta.in.CI_Z_w_shared_ad[i] <- "yes"
}




if (C == "CRn") {
  if (lower_Z_g_shared_mu < true_beta & true_beta < upper_Z_g_shared_mu) {
    true.beta.in.CI_Z_g_shared_mu[i] <- "yes"
  }
  if (lower_Z_g_shared_ad < true_beta & true_beta < upper_Z_g_shared_ad) {
    true.beta.in.CI_Z_g_shared_ad[i] <- "yes"
  }
}

if (C == "CRDP") {
  if (lower_Z_c_shared_mu < true_beta & true_beta < upper_Z_c_shared_mu) {
    true.beta.in.CI_Z_c_shared_mu[i] <- "yes"
```

```
    }
    if (lower_Z_c_shared_ad < true_beta & true_beta < upper_Z_c_shared_ad) {
      true.beta.in.CI_Z_c_shared_ad[i] <- "yes"
    }
  }

  if (lower_Z_f_shared_mu < true_beta & true_beta < upper_Z_f_shared_mu) {
    true.beta.in.CI_Z_f_shared_mu[i] <- "yes"
  }
  if (lower_Z_f_shared_ad < true_beta & true_beta < upper_Z_f_shared_ad) {
    true.beta.in.CI_Z_f_shared_ad[i] <- "yes"
  }

}

if (C == "CRn") {
  result <- list(coef_X = coef_X,
                 coef_Z_w_unshared_mu = coef_Z_w_unshared_mu,
                 coef_Z_w_unshared_ad = coef_Z_w_unshared_ad,
                 coef_Z_g_unshared_mu = coef_Z_g_unshared_mu,
                 coef_Z_g_unshared_ad = coef_Z_g_unshared_ad,
                 coef_Z_f_unshared_mu = coef_Z_f_unshared_mu,
                 coef_Z_f_unshared_ad = coef_Z_f_unshared_ad,
                 coef_Z_w_shared_mu = coef_Z_w_shared_mu,
                 coef_Z_w_shared_ad = coef_Z_w_shared_ad,
                 coef_Z_g_shared_mu = coef_Z_g_shared_mu,
                 coef_Z_g_shared_ad = coef_Z_g_shared_ad,
                 coef_Z_f_shared_mu = coef_Z_f_shared_mu,
                 coef_Z_f_shared_ad = coef_Z_f_shared_ad,
                 age_mean = age_mean,
                 age_median = age_median,
                 true.beta.in.CI_X = true.beta.in.CI_X,
                 true.beta.in.CI_Z_w_unshared_mu = true.beta.in.CI_Z_w_unshared_mu,
                 true.beta.in.CI_Z_w_unshared_ad = true.beta.in.CI_Z_w_unshared_ad,
                 true.beta.in.CI_Z_g_unshared_mu = true.beta.in.CI_Z_g_unshared_mu,
                 true.beta.in.CI_Z_g_unshared_ad = true.beta.in.CI_Z_g_unshared_ad,
                 true.beta.in.CI_Z_f_unshared_mu = true.beta.in.CI_Z_f_unshared_mu,
                 true.beta.in.CI_Z_f_unshared_ad = true.beta.in.CI_Z_f_unshared_ad,
                 true.beta.in.CI_Z_w_shared_mu = true.beta.in.CI_Z_w_shared_mu,
                 true.beta.in.CI_Z_w_shared_ad = true.beta.in.CI_Z_w_shared_ad,
                 true.beta.in.CI_Z_g_shared_mu = true.beta.in.CI_Z_g_shared_mu,
                 true.beta.in.CI_Z_g_shared_ad = true.beta.in.CI_Z_g_shared_ad,
                 true.beta.in.CI_Z_f_shared_mu = true.beta.in.CI_Z_f_shared_mu,
                 true.beta.in.CI_Z_f_shared_ad = true.beta.in.CI_Z_f_shared_ad
                 )
}

if (C == "CRDP") {
  result <- list(coef_X = coef_X,
                 coef_Z_w_unshared_mu = coef_Z_w_unshared_mu,
                 coef_Z_w_unshared_ad = coef_Z_w_unshared_ad,
                 coef_Z_c_unshared_mu = coef_Z_c_unshared_mu,
                 coef_Z_c_unshared_ad = coef_Z_c_unshared_ad,
```

```
                  coef_Z_f_unshared_mu = coef_Z_f_unshared_mu,
                  coef_Z_f_unshared_ad = coef_Z_f_unshared_ad,
                  coef_Z_w_shared_mu = coef_Z_w_shared_mu,
                  coef_Z_w_shared_ad = coef_Z_w_shared_ad,
                  coef_Z_c_shared_mu = coef_Z_c_shared_mu,
                  coef_Z_c_shared_ad = coef_Z_c_shared_ad,
                  coef_Z_f_shared_mu = coef_Z_f_shared_mu,
                  coef_Z_f_shared_ad = coef_Z_f_shared_ad,
                  age_mean = age_mean,
                  age_median = age_median,
                  true.beta.in.CI_X = true.beta.in.CI_X,
                  true.beta.in.CI_Z_w_unshared_mu = true.beta.in.CI_Z_w_unshared_mu,
                  true.beta.in.CI_Z_w_unshared_ad = true.beta.in.CI_Z_w_unshared_ad,
                  true.beta.in.CI_Z_c_unshared_mu = true.beta.in.CI_Z_c_unshared_mu,
                  true.beta.in.CI_Z_c_unshared_ad = true.beta.in.CI_Z_c_unshared_ad,
                  true.beta.in.CI_Z_f_unshared_mu = true.beta.in.CI_Z_f_unshared_mu,
                  true.beta.in.CI_Z_f_unshared_ad = true.beta.in.CI_Z_f_unshared_ad,
                  true.beta.in.CI_Z_w_shared_mu = true.beta.in.CI_Z_w_shared_mu,
                  true.beta.in.CI_Z_w_shared_ad = true.beta.in.CI_Z_w_shared_ad,
                  true.beta.in.CI_Z_c_shared_mu = true.beta.in.CI_Z_c_shared_mu,
                  true.beta.in.CI_Z_c_shared_ad = true.beta.in.CI_Z_c_shared_ad,
                  true.beta.in.CI_Z_f_shared_mu = true.beta.in.CI_Z_f_shared_mu,
                  true.beta.in.CI_Z_f_shared_ad = true.beta.in.CI_Z_f_shared_ad
                  )
  }


  return(result)
}
```

### Data2, CRn

```
set.seed(12345)

data2_radon <- data2
data2_radon$X <- NULL

data2_radon[, `:=` (mine = sample(1:5, 1, replace = T),
                    shaft = sample(1:5, 1, replace = T)),
            by = ID]

data2_radon[, `:=` (
  CRn = sample(x = CRn$CRn, 1, replace = T, prob = CRn$p_appr)),
  by = c("year", "mine")]

data2_radon[, f := sample(activity_weighting_factors, 1, replace = T),
            by = ID]

data2_radon[, `:=` (w = sample(working_time_factors, nrow(data2_radon), replace = T),
                    g = sample(equilibrium_factors, nrow(data2_radon), replace = T))]

data2_radon[, `:=` (w_mean = mean(w),
```

```
                    g_mean = mean(g)),
            by = c("year", "mine")]

data2_radon[, f_mean := mean(f), by = c("year", "mine", "shaft")]

data2_radon[, `:=` (
  U_w_unshared_mu = assignment_u(var = 2 * var(working_time_factors), nrow(data2_radon),
                                 multiplicative = T),
  U_w_unshared_ad = assignment_u(var = var(working_time_factors), nrow(data2_radon),
                                 multiplicative = F),
  U_g_unshared_mu = assignment_u(var = 2 * var(equilibrium_factors), nrow(data2_radon),
                                 multiplicative = T),
  U_g_unshared_ad = assignment_u(var = var(equilibrium_factors), nrow(data2_radon),
                                 multiplicative = F),
  U_f_unshared_mu = assignment_u(var = 2 * var(activity_weighting_factors), nrow(data2_radon),
                                 multiplicative = T),
  U_f_unshared_ad = assignment_u(var = var(activity_weighting_factors), nrow(data2_radon),
                                 multiplicative = F))]

data2_radon[, `:=` (
  U_w_shared_mu = assignment_u(var = 2 * var(working_time_factors), 1, multiplicative = T),
  U_w_shared_ad = assignment_u(var = var(working_time_factors), 1, multiplicative = F),
  U_g_shared_mu = assignment_u(var = 2 * var(equilibrium_factors), 1, multiplicative = T),
  U_g_shared_ad = assignment_u(var = var(equilibrium_factors), 1, multiplicative = F),
  U_f_shared_mu = assignment_u(var = 2 * var(activity_weighting_factors), 1, multiplicative = T),
  U_f_shared_ad = assignment_u(var = var(activity_weighting_factors), 1, multiplicative = F)),
  by = c("mine", "year")]

data2_radon[, `:=` (
  estim_w_unshared_mu = w_mean * U_w_unshared_mu,
  estim_w_unshared_ad = w_mean + U_w_unshared_ad,
  estim_g_unshared_mu = g_mean * U_g_unshared_mu,
  estim_g_unshared_ad = g_mean + U_g_unshared_ad,
  estim_f_unshared_mu = f_mean * U_f_unshared_mu,
  estim_f_unshared_ad = f_mean + U_f_unshared_ad,

  estim_w_shared_mu = w_mean * U_w_shared_mu,
  estim_w_shared_ad = w_mean + U_w_shared_ad,
  estim_g_shared_mu = g_mean * U_g_shared_mu,
  estim_g_shared_ad = g_mean + U_g_shared_ad,
  estim_f_shared_mu = f_mean * U_f_shared_mu,
  estim_f_shared_ad = f_mean + U_f_shared_ad)]

data2_radon[, `:=` (
  X = CRn * 12 * f * w * g,

  Z_w_unshared_mu = CRn * 12 * f * estim_w_unshared_mu * g,
  Z_w_unshared_ad = CRn * 12 * f * estim_w_unshared_ad * g,
  Z_g_unshared_mu = CRn * 12 * f * w * estim_g_unshared_mu,
  Z_g_unshared_ad = CRn * 12 * f * w * estim_g_unshared_ad,
  Z_f_unshared_mu = CRn * 12 * estim_f_unshared_mu * w * g,
  Z_f_unshared_ad = CRn * 12 * estim_f_unshared_ad * w * g,
```

```
  Z_w_shared_mu = CRn * 12 * f * estim_w_shared_mu * g,
  Z_w_shared_ad = CRn * 12 * f * estim_w_shared_ad * g,
  Z_g_shared_mu = CRn * 12 * f * w * estim_g_shared_mu,
  Z_g_shared_ad = CRn * 12 * f * w * estim_g_shared_ad,
  Z_f_shared_mu = CRn * 12 * estim_f_shared_mu * w * g,
  Z_f_shared_ad = CRn * 12 * estim_f_shared_ad * w * g)]



data2_radon[, Xcum_100 := cumsum(X)/100, by = ID]

cols <- names(data2_radon)[stringr::str_detect(names(data2_radon), "Z")]

data2_radon[, (paste0("Zcum_100", substr(cols, 2, 20))) :=
              lapply(.SD, function(x) cumsum(x)/100), .SDcols = cols, by = ID]

setorder(data2_radon, ID, year)


result_data2_estim_delta <- simulate_delta_estim(data = data2_radon)
saveRDS(result_data2_estim_delta, "estim/result_data2_estim_delta.RDS")

result_data2_estim_delta_censor <- simulate_delta_estim(data = data2_radon, censor = 0.1)
saveRDS(result_data2_estim_delta_censor, "estim/result_data2_estim_delta_censor.RDS")

result_data2_estim_sim <- simulate_study_estim(result_data2_estim_delta, C = "CRn")
saveRDS(result_data2_estim_sim, "estim/result_data2_estim_sim.RDS")

result_data2_estim_sim_censor <- simulate_study_estim(result_data2_estim_delta_censor, C = "CRn")
saveRDS(result_data2_estim_sim_censor, "estim/result_data2_estim_sim_censor.RDS")
```

## Data1, CRDP

```
set.seed(12345)

data1_radon <- data1

data1_radon[, CRDP := X / 12 / mean(activity_weighting_factors) /
              mean(working_time_factors) /  mean(correction_factors)]

data1_radon[, `:=` (mine = sample(1:5, 1, replace = T),
                    shaft = sample(1:5, 1, replace = T)),
                    by = ID]

data1_radon[, CRDP_mean := mean(CRDP), by = c("year", "mine")]

data1_radon[, f := sample(activity_weighting_factors, 1, replace = T), by = ID]

data1_radon[, `:=` (w = sample(working_time_factors, nrow(data1_radon), replace = T),
                    c = sample(correction_factors, nrow(data1_radon), replace = T))]

data1_radon[, w_mean := mean(w), by = c("year", "mine")]
```

```
data1_radon[, f_mean := mean(f), by = c("year", "mine", "shaft")]

data1_radon[, c_mean := mean(c), by = c("mine")]


data1_radon[, `:=` (
  U_w_unshared_mu = assignment_u(var = 2 * var(working_time_factors), nrow(data1_radon),
                                 multiplicative = T),
  U_w_unshared_ad = assignment_u(var = var(working_time_factors), nrow(data1_radon),
                                 multiplicative = F),
  U_c_unshared_mu = assignment_u(var = 2 * var(correction_factors), nrow(data1_radon),
                                 multiplicative = T),
  U_c_unshared_ad = assignment_u(var = var(correction_factors), nrow(data1_radon),
                                 multiplicative = F),
  U_f_unshared_mu = assignment_u(var = 2 * var(activity_weighting_factors), nrow(data1_radon),
                                 multiplicative = T),
  U_f_unshared_ad = assignment_u(var = var(activity_weighting_factors), nrow(data1_radon),
                                 multiplicative = F))]

data1_radon[, `:=` (
  U_w_shared_mu = assignment_u(var = 2 * var(working_time_factors), 1, multiplicative = T),
  U_w_shared_ad = assignment_u(var = var(working_time_factors), 1, multiplicative = F),
  U_c_shared_mu = assignment_u(var = 2 * var(correction_factors), 1, multiplicative = T),
  U_c_shared_ad = assignment_u(var = var(correction_factors), 1, multiplicative = F),
  U_f_shared_mu = assignment_u(var = 2 * var(activity_weighting_factors), 1, multiplicative = T),
  U_f_shared_ad = assignment_u(var = var(activity_weighting_factors), 1, multiplicative = F)),
  by = c("mine", "year")]

data1_radon[, `:=` (
  estim_w_unshared_mu = w_mean * U_w_unshared_mu,
  estim_w_unshared_ad = w_mean + U_w_unshared_ad,
  estim_c_unshared_mu = c_mean * U_c_unshared_mu,
  estim_c_unshared_ad = c_mean + U_c_unshared_ad,
  estim_f_unshared_mu = f_mean * U_f_unshared_mu,
  estim_f_unshared_ad = f_mean + U_f_unshared_ad,

  estim_w_shared_mu = w_mean * U_w_shared_mu,
  estim_w_shared_ad = w_mean + U_w_shared_ad,
  estim_c_shared_mu = c_mean * U_c_shared_mu,
  estim_c_shared_ad = c_mean + U_c_shared_ad,
  estim_f_shared_mu = f_mean * U_f_shared_mu,
  estim_f_shared_ad = f_mean + U_f_shared_ad)]

data1_radon[, `:=` (
  X = CRDP_mean * 12 * f * w * c,

  Z_w_unshared_mu = CRDP_mean * 12 * f * estim_w_unshared_mu * c,
  Z_w_unshared_ad = CRDP_mean * 12 * f * estim_w_unshared_ad * c,
  Z_c_unshared_mu = CRDP_mean * 12 * f * w * estim_c_unshared_mu,
  Z_c_unshared_ad = CRDP_mean * 12 * f * w * estim_c_unshared_ad,
  Z_f_unshared_mu = CRDP_mean * 12 * estim_f_unshared_mu * w * c,
  Z_f_unshared_ad = CRDP_mean * 12 * estim_f_unshared_ad * w * c,
```

```
  Z_w_shared_mu = CRDP_mean * 12 * f * estim_w_shared_mu * c,
  Z_w_shared_ad = CRDP_mean * 12 * f * estim_w_shared_ad * c,
  Z_c_shared_mu = CRDP_mean * 12 * f * w * estim_c_shared_mu,
  Z_c_shared_ad = CRDP_mean * 12 * f * w * estim_c_shared_ad,
  Z_f_shared_mu = CRDP_mean * 12 * estim_f_shared_mu * w * c,
  Z_f_shared_ad = CRDP_mean * 12 * estim_f_shared_ad * w * c)]



data1_radon[, Xcum_100 := cumsum(X)/100, by = ID]

cols <- names(data1_radon)[stringr::str_detect(names(data1_radon), "Z")]

data1_radon[, (paste0("Zcum_100", substr(cols, 2, 20))) :=
              lapply(.SD, function(x) cumsum(x)/100), .SDcols = cols, by = ID]

setorder(data1_radon, ID, year)


result_data1_estim_delta <- simulate_delta_estim(data = data1_radon)
saveRDS(result_data1_estim_delta, "estim/result_data1_estim_delta.RDS")

result_data1_estim_delta_censor <- simulate_delta_estim(data = data1_radon, censor = 0.1)
saveRDS(result_data1_estim_delta_censor, "estim/result_data1_estim_delta_censor.RDS")

result_data1_estim_sim <- simulate_study_estim(result_data1_estim_delta, C = "CRDP")
saveRDS(result_data1_estim_sim, "estim/result_data1_estim_sim.RDS")

result_data1_estim_sim_censor <- simulate_study_estim(result_data1_estim_delta_censor, C = "CRDP")
saveRDS(result_data1_estim_sim_censor, "estim/result_data1_estim_sim_censor.RDS")
```

# General information

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Mojave 10.14.6
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] de_DE.UTF-8/de_DE.UTF-8/de_DE.UTF-8/C/de_DE.UTF-8/de_DE.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] survival_2.44-1.1 msm_1.6.7         doParallel_1.0.15 iterators_1.0.12
## [5] foreach_1.4.7     dplyr_0.8.3       data.table_1.12.2
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.2       knitr_1.24       magrittr_1.5     splines_3.6.1
##  [5] tidyselect_0.2.5 lattice_0.20-38  R6_2.4.0         rlang_0.4.0
##  [9] stringr_1.4.0    tools_3.6.1      grid_3.6.1       xfun_0.9
## [13] htmltools_0.3.6  yaml_2.2.0       assertthat_0.2.1 digest_0.6.20
## [17] tibble_2.1.3     crayon_1.3.4     Matrix_1.2-17    purrr_0.3.2
## [21] codetools_0.2-16 glue_1.3.1       evaluate_0.14    rmarkdown_1.15
## [25] stringi_1.4.3    compiler_3.6.1   pillar_1.4.2     expm_0.999-4
## [29] mvtnorm_1.0-11   pkgconfig_2.0.2
```